Coding Summary By Code
SLR
31/03/2022 13:09

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On

Node

Nodes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications\Architecture of ML system for maintainability

PDF

Files\\AI lifecycle models need to be revised

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On
No	Google Scholar	0.0344	10			
				1	S	10/02/2022 11:57
				2	S	10/02/2022 11:57
				3	S	10/02/2022 11:58

Moreover, model risk experts are now required to have a strong background in two disjoints fields: 1) Governance, Risk Management, and Compliance and 2) AI. We conjecture

				4	S	10/02/2022 11:59

Technology Access All AI technologies, tools, and libraries need to be audited to make
sure they are safe to be used in fintech applications. Only then, practitioners are able to design their Machine Learning systems around the latest technology. This is a challenge that needs to tackled by any organization akin to ING. As presented in Section 4.4, this process can be limiting since new AI technologies are appearing every day. Practitioners willing to try the latest AI technology may feel less motivated since it may take some time before they are approved. As referred in Section 4.1, many problems at ING are triggered by the Technology push. Hence, new business opportunities might be missed if practitioners are not able to experiment the latest AI technologies. We do not know to what extent Technology Access is also a challenge to software orga-
nizations operating in other domains. Previous work suggests that only 8% of software developers consider an organization’s culture and policies highly-influential when selecting third-party software libraries (Larios Vargas et al. 2020).

				5	S	10/02/2022 11:59

6.1 Implications We see the following implications of our results for the fintech industry and for research. 6.1.1 Implications for Machine Learning Practitioners
Machine Learning practitioners have to be aware of extra steps and challenges in their process of developing Machine Learning applications. Although not mentioned in existing lifecycle models, the undertaking of feasibility assessments, documentation, and model monitoring, are crucial while developing Machine Learning applications.

				6	S	10/02/2022 11:59

6.1.2 Implications for Process Architects
Existing lifecycle models provide a canonical overview of the multiple stages in the lifecycle of a Machine Learning application. However, when being applied to a particular context, such as fintech, these models need to be adapted. From our findings, we suspect that this is also the case for other fields where AI is getting increasing importance.

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On
				7	S	10/02/2022 11:59

6.1.3 Implications for Researchers
Researchers could focus on solving the reported challenges in the Machine Learning lifecycle with additional tool support and reveal challenges of the ML lifecycle in other domains by extending the case study to more organizations and different types of industries. More automation is required for exploratory data analysis and data integration tech-
niques (Mitchell et al. 2019; Damiani and Frati 2018). Moreover, there are minimal advancements in documentation of Machine Learning projects. Techniques ought to be studied to help trace documentation back to the codebase and vice versa.

				8	S	10/02/2022 11:59

6.1.4 Implications for Tool Developers
Although a number of tools are emerging to aid ML engineering, these solutions fail to address the singularities of different projects. Thus, practitioners are adopting their own customized solutions. For example, spreadsheets are still being used to manually log experiments regardless of the existing automated solutions, such as MLFlow, DVC, Replicate, and so on. It is important to understand what is missing in the current solutions and how we can propose a solution that effectively solves version control to keep track of changes in data, changes in scoring metrics, and executions of different experiments.

				9	S	10/02/2022 11:59

6.1.5 Implications for Educators
Page 25 of 29 95
Education of Machine Learning should focus on the whole lifecycle of Machine Learning development, including exploratory analysis with a focus on statistics, data analysis and data visualization. Moreover, practitioners with background on both data science and software engineering are a valuable resource for organizations.

				10	S	10/02/2022 12:00

6.1.6 Implications for Organizations Embracing AI
The embrace of AI stretches the adequacy of well-established processes at organizations. Multi-disciplinary teams are essential to embrace AI: AI experts have the knowledge to try innovative approaches, but will likely have little expertise to identify business value. Thus, knowledge transfer between stakeholders is challenging and might hinder the motivation of developers. New strategies must be outlined to reduce the amount of effort required to document AI projects.

Files\\An End-to-End Framework for Productive Use of Machine Learning in Software Analytics and Business Intelligence Solutions

No	Google Scholar	0.0277	3			
				1	S	09/02/2022 13:23

The framework is structured in three iterative cycles representing different stages in a model’s lifecycle: prototyping, deployment, update. As a result, the framework specifically supports the transitions between these stages while also covering all important activities from data collection to retraining deployed ML models. To validate the applicability of the framework in practice, we compare it to and apply it in a real-world ML-based SA/BI solution.

				2	S	09/02/2022 13:39

While the literature review examines the topics data management and pro-
cessing, model building,and model deployment and serving individually, in reality a separation of the three is not that trivial. In fact, for building end-to-end solutions the fields are very much interrelated as the activities depend on each other and sometimes even overlap. Oftentimes, ML projects start out as a prototypical analysis due to a lim-
ited amount of time and resources [15,30]. In order to use and actually benefit from the ML model, it needs to be deployed to a production environment which can be time and cost-intensive but nonetheless crucial [21,30]. To avoid the deployed models from being outdated, it is important to provide a functionality for dynamically deploying new models or iteratively retraining and updating existing models [9,21]. As a result, we identify three iterative cycles which are passed through during
an end-to-end development of ML solutions and, therefore, serve as the main dimensions in our framework: 1) Prototyping cycle (blue), 2) deployment cycle (green), and 3) update cycle (orange).

				3	S	09/02/2022 13:35

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On

Files\\Applying AI in Practice~ Key Challenges and Lessons Learned

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On
No	Scopus	0.0313	9			
				1	S	08/02/2022 13:55

Approaches, In-Progress Research and Lessons Learned
In this section we discuss ongoing research facing the outlined challenges in the previous section, comprising:
(1) Automated and Continuous Data Quality Assurance, see Sect. 3.1; (2) Domain Adaptation Approach for Tackling Deviating Data Characteristics at Training and Test Time, see Sect. 3.2;
(3) Hybrid Model Design for Improving Model Accuracy, see Sect. 3.3;

2	S	08/02/2022 13:55

(4) Interpretability by Correction Model Approach, see Sect. 3.4; (5) Software Quality by Automated Code Analysis and Documentation Generation, see Sect. 3.5;
(6) The ALOHA Toolchain for Embedded Platforms, see Sect. 3.6; (7) Human AI Teaming as Key to Human Centered AI, see Sect. 3.7.

3	S	08/02/2022 13:56

Approach 1: Automated and Continuous Data Quality Assurance
In times of large and volatile amounts of data, which are often generated automatically by sensors (e.g., in smart home solutions of housing units or industrial settings), it is especially important to, (i), automatically, and, (ii), continuously monitor the quality of data [22,88]. A recent study [20] shows that the continuous monitoring of data quality is only supported by very few

4	S	08/02/2022 13:56

Approach 2: The Domain Adaptation Approach for Tackling Deviating Data Characteristics at Training and Test Time
In [106] and [108] we introduce a novel distance measure, the so-called Centralized Moment Discrepancy (CMD), for aligning probability distributions in the context of domain adaption. Domain adaptation algorithms are designed to minimize the misclassification risk of a discriminative

5	S	08/02/2022 13:56

Approach 3: Hybrid Model Design for Improving Model Accuracy by Integrating Expert Hints in Biomedical Diagnostics
For diagnostics based on biomedical image analysis, image segmentation serves as a prerequisite step to extract quantitative information [70]. If, however, segmentation results are not accurate, quantitative analysis can lead to results

6	S	08/02/2022 13:57

Approach 4: Interpretability by Correction Model Approach
Last year, at a symposium on predictive analytics in Vienna [93], we introduced an approach to the problem of formulating interpretability of AI models for classification or regression problems [37] with a given basis model, e.g., in the context of model predictive control [32]. The basic idea is to root the problem of interpretability in the basic model by considering the contribution of the AI model as correction of this basis model and is referred to as “Before and After Correction Parameter Comparison (BAPC)”. The idea

7	S	08/02/2022 13:57

Approach 5: Software Quality by Code Analysis and Automated Documentation
Quality assurance measures in software engineering include, e.g., automated testing [2], static code analysis [73], system redocumentation [69], or symbolic execution [4]. These measures need to be risk-based [23,83], exploiting knowledge about system and design dependencies, business requirements, or characteristics of the applied development process.

8	S	08/02/2022 13:57

Approach 6: The ALOHA Toolchain for Embedded Platforms
In [66] and [65] we introduce ALOHA, an integrated tool flow that tries to make the design of deep learning (DL) applications and their porting on embedded heterogeneous architectures as simple and painless as possible. ALOHA is the result of interdisciplinary research funded by the EU13.

9	S	08/02/2022 13:57

Approach 7: Human AI Teaming Approach as Key to Human Centered AI
In [36], we introduce an approach for human-centered AI in working environments utilizing knowledge graphs and relational machine learning ([72,79]). This approach is currently being refined in the ongoing Austrian project Humancentred AI in digitised working environments (AI@Work). The discussion starts with a critical analysis of the limitations of current AI systems whose learning/training is restricted to predefined structured data, most vector-based with a pre-defined format.

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On

Files\\Large-scale machine learning systems in real-world industrial settings~ A review of challenges and solutions

No	Web of science	0.0166	2			
				1	S	11/02/2022 14:20

Background : Developing and maintaining large scale machine learning (ML) based software systems in an industrial setting is challenging. There are no well-established development guidelines, but the literature contains reports on how companies develop and maintain deployed ML-based software systems. Objective : This study aims to survey the literature related to development and maintenance of large scale MLbased systems in industrial settings in order to provide a synthesis of the challenges that practitioners face. In addition, we identify solutions used to address some of these challenges. Method : A systematic literature review was conducted and we identified 72 papers related to development and maintenance of large scale ML-based software systems in industrial settings. The selected articles were qualitatively analyzed by extracting challenges and solutions. The challenges and solutions were thematically synthesized into four quality attributes: adaptability, scalability, safety and privacy. The analysis was done in relation to ML workflow, i.e. data acquisition, training, evaluation, and deployment. Results : We identified a total of 23 challenges and 8 solutions related to development and maintenance of large scale ML-based software systems in industrial settings including six different domains. Challenges were most often reported in relation to adaptability and scalability. Safety and privacy challenges had the least reported solutions.

				2	S	11/02/2022 14:22

Files\\Overton~ A Data System for Monitoring and Improving Machine-Learned Products

No	Google Scholar	0.0453	3			
				1	S	24/02/2022 09:56

Overton provides the engineer with abstractions that allow them to build, maintain, and monitor their application
by manipulating data files–not custom code. Inspired by relational systems, supervision (data) is managed separately from the model (schema). Akin to traditional logical independence, Overton's schema provides model independence: serving code does not change even when inputs, parameters, or resources of the model change. The schema changes very infrequently–many production services have not updated their schema in over a year. Overton takes as input a schema whose design goal is to support rich applications from modeling to automatic
deployment. In more detail, the schema has two elements: (1) data payloads similar to a relational schema, which describe the input data, and (2) model tasks, which describe the tasks that need to be accomplished. The schema defines the input, output, and coarse-grained data flow of a deep learning model. Informally, the schema defines what the model computes but not how the model computes it: Overton does not prescribe architectural details of the underlying model (e.g., Overton is free to embed sentences using an LSTM or a Transformer) or hyperparameters, like hidden state size. Additionally, sources of supervision are described as data–not in the schema–so they are free to rapidly evolve. As shown in Figure 1, given a schema and a data file, Overton is responsible to instantiate and train a model,
combine supervision, select the model's hyperparameters, and produce a production-ready binary. Overton compiles the schema into a (parameterized) TensorFlow or PyTorch program, and performs an architecture and hyperparameter search. A benefit of this compilation

				2	S	24/02/2022 09:56

(1) Code-free Deep Learning In Overton-based systems, engineers focus exclusively on fine-grained monitoring
of their application quality and improving supervision–not tweaking deep learning models. An Overton engineer does

	Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On
					3	S	24/02/2022 09:56

technique led to state-of-the-art results on natural language benchmarks including GLUE and SuperGLUE [31].3 (2) Multitask Learning Overton was built to natively support multitask learning [2,24,26] so that all model tasks
are concurrently predicted. A key benefit is that Overton can accept supervision at whatever granularity (for whatever task) is available. Overton models often perform ancillary tasks like part-of-speech tagging or typing. Intuitively, if a representation has captured the semantics of a query, then it should reliably perform these ancillary tasks. Typically, ancillary tasks are also chosen either to be inexpensive to supervise. Ancillary task also allow developers to gain confidence in the model's predictions and have proved to be helpful for aids for debugging errors. (3) Weak Supervision Applications have access to supervision of varying quality and combining this contradictory
and incomplete supervision is a major challenge. Overton uses techniques from Snorkel [23] and Google's Snorkel DryBell [12], which have studied how to combine supervision in theory and in software. Here, we describe two novel observations from building production applications: (1) we describe the shift to applications which are constructed almost entirely with weakly supervised data due to cost, privacy, and cold-start issues, and (2) we observe that weak supervision may obviate the need for popular methods like transfer learning from massive pretrained models, e.g., BERT [8]–on some production workloads, which suggests that a deeper trade-off study may be

Nodes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications\Auto ML

PDF

Files\\A Meta Learning Approach for Automating Model Selection in Big Data Environments using Microservice and Container Virtualization Technologies

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On
No	ACM Digital library	0.0281	2			
				1	S	11/02/2022 12:44

In the following, some meta learning approaches developed as
frameworks with wizard [12][26][8][5][11] are discussed. A parallelized, component-based, modular and easily extendable meta learning system for univariate and multivariate time series load forecasting can be found in [12]. Matijaš et. al. [12] built the meta learner as an ensemble method. As meta features, minimum, maximum, Standard Deviation (SD), skewness, to name a few, were considered. Auto-WEKA [26] is a framework for automatically selecting classi�ers and hyperparameters implemented in WEKA. In the updated version Auto -WEKA 2.0 [8], they also supported regression algorithms and a more tightly integration with WEKA. Auto-Sklearn [5] is a meta learning framework based on scikitlearn which uses the same principles as Auto-WEKA. To solve the Combined Algorithm Selection and Hyperparameter optimization (CASH) problem, they built on the research from Auto-WEKA and used the same Sequential Model based Algorithm Con�guration (SMAC) algorithm as Bayesian optimizer for hyperparameter tuning. The drawback in Auto-WEKA and Auto-Sklearn is that they are implemented as monolithic applications which limit the scalability and increase the di�culty of maintenance. Moreover, they did not provide the possibility to handle model selection for large amount of data. SmartML [11] is a meta learning framework based on the R language. It is implemented as web application with REST APIs. SmartML can recommend a classi�cation algorithm, including hyperparameter tuning based on a total of 25 meta features. The limitation here is also that SmartML does not support Big Data environment for large scale processing. In contrast to the aforementioned works, the current framework in the present paper is implemented as a microservice architecture to increase the scalability and facilitate maintainability. Moreover, the utilization of a powerful Big Data stack gives the ability to perform model selection for large amount of data.

			2	S	11/02/2022 12:44

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On

Files\\Auto-Keras~ An Efficient Neural Architecture Search System

No	Google Scholar	0.0332	6			
				1	S	08/02/2022 13:42

Neural architecture search (NAS) has been proposed to automat-
ically tune deep neural networks, but existing search algorithms, e.g., NASNet [51], PNAS [29], usually suffer from expensive computational cost. Network morphism, which keeps the functionality of a neural network while changing its neural architecture, could be helpful for NAS by enabling more efficient training during the search. In this paper, we propose a novel framework enabling Bayesian optimization to guide the network morphism for efficient neural architecture search. The framework develops a neural network kernel and a tree-structured acquisition function optimization algorithm to efficiently explores the search space. Extensive experiments on real-world benchmark datasets have been done to demonstrate the superior performance of the developed framework over the state-of-the-

				2	S	08/02/2022 13:42

Automated Machine Learning (AutoML) has become a very important research topic with wide applications of machine learning techniques. The goal ofAutoML is to enable people with limited machine learning background knowledge to use machine learning models easily. Work has been done on automated model selection, automated hyperparameter tunning, and etc. In the context of deep

				3	S	08/02/2022 13:43

learning, neural architecture search (NAS), which aims to search for the best neural network architecture for the given learning task and dataset, has become an effective computational tool in AutoML. Unfortunately, existing NAS algorithms are usually computationally expensive. The time complexity ofNAS is O(n¯t), where n is the number of neural architectures evaluated during the search, and ¯t is the average time consumption for evaluating each of the n neural networks. Many NAS approaches, such as deep reinforcement learning [2, 37, 47, 50, 51], gradient-based methods [8, 31, 33] and evolutionary algorithms [12, 17, 30, 38, 39, 41], require a large n to reach a good performance. Moreover, many of them train each of the n neural networks from scratch, which is very slow.

				4	S	08/02/2022 13:44

In addition, we have developed a widely adopted open-source
AutoML system based on our proposed method, namely Auto-Keras. It is an open-source AutoML system, which can be download and installed locally. The system is carefully designed with a concise interface for people not specialized in computer programming and data science to use. To speed up the search, the workload on CPU and GPU can run in parallel. To address the issue of different GPU memory, which limits the size ofthe neural architectures, a memory adaption strategy is designed for deployment. The main contributions of the paper are as follows:

• Propose an algorithm for efficient neural architecture search based on network morphism guided by Bayesian optimization.

• Conduct intensive experiments on benchmark datasets to demonstrate the superior performance of the proposed method over the baseline methods.

				5	S	08/02/2022 13:44

Based on the proposed neural architecture search method, we developed an open-source AutoML system, namely Auto-Keras. It is named after Keras [11], which is known for its simplicity in creating neural networks. Similar to SMAC [21], TPOT [35], AutoWEKA [44], and Auto-Sklearn [15], the goal is to enable domain experts who are not familiar with machine learning technologies to use machine learning techniques easily. However, Auto-Keras is focusing on the deep learning tasks, which is different from the systems focusing on the shallow models mentioned above.

				6	S	08/02/2022 13:45

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On

Files\\Autonomic machine learning platform

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On
No	Google Scholar	0.0979	7			
				1	S	11/02/2022 14:50

Acquiring information properly through machine learning requires familiarity with the available algorithms and understanding how they work and how to address the given problem in the best possible way. However, even for machine-learning experts in specific industrial fields, in order to predict and acquire information properly in different industrial fields, it is necessary to attempt several instances of trial and error to succeed with the application of machine learning. For non-experts, it is much more difficult to make accurate predictions through machine learning. In this paper, we propose an autonomic machine learning platform which provides the decision factors to be made during the developing of machine learning applications. In the proposed autonomic machine learning platform, machine learning processes are automated based on the specification of autonomic levels. This autonomic machine learning platform can be used to derive a high-quality learning result by minimizing experts' interventions and reducing the number of design selections that require expert knowledge and intuition. We also demonstrate that the proposed autonomic machine learning platform is suitable for smart cities which typically require considerable amounts of security sensitive information.

Reference Number	Coded By Initials	Modified On
2	S	11/02/2022 14:51
3	S	11/02/2022 14:51
4	S	11/02/2022 14:51
5	S	11/02/2022 14:52
6	S	11/02/2022 14:52
7	S	11/02/2022 14:52

This study aims to present the need of the autonomic machine
learning platform for the universal use of machine learning techniques in a variety of applications including Smart City, Smart Factory, and Smart Grid. This study has several unique contributions and implications, given
as follow:

• This study presents twelve design factors to be required by expert knowledge and intuition during the machine learning development process.

• This study defines five levels of autonomic machine learning referring to as the degree of expert interventions based on the steps of the machine learning development process.

• The levels of autonomic machine learning can minimize expert intervention at various autonomic levels by reducing the number of design selections that require expert knowledge and intuition is proposed. This autonomic machine learning platform can be used to derive a high-quality learning result.

• This study focuses on the design issues in terms of the practical autonomic machine learning by applying the autonomic machine learning related to smart cities from an information systems perspective. Therefore, this study is useful for system developers involved in smart city development initiatives using machine learning.

• In a truly smart city of the future, automation will be paramount to improve the service level of the end users (Rana et al., 2018). This capability can be derived from advanced information technologies such as the proposed autonomic machine learning platform.

Like any publication, this study has certain limitations, given as follow:

• This study only focuses the design of the autonomous machine learning platform, but the actual implementation or application may be very different from the proposed design structure and it does not cover issues related to implementing the autonomic machine learning techniques and systems.

• As a research study, no primary data was collected or used to support the development of the proposed autonomic machine learning platform.

• This study could not provide a valuable synthesis of the relevant literature by analyzing and discussing the key findings from the existing

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On

Files\\AutoTrain~ An Efficient Auto-training System for Small-scale Image Classification

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On
No	IEEE	0.0729	6			
				1	S	08/02/2022 13:12

In this paper, we propose an efficient automatic training system, AutoTrain, to solve small-scale image classification problems. First, we design sample equalization in data augmentation to improve the performance of training on uneven data. Then, a Bayesian optimization-based strategy controller is introduced to rapidly find the strategy applied in data augmentation. Additionally, we present a dynamic adjustment model to fit tasks with different scales and complexity. Finally, experimental results show that the AutoTrain's training speed is about 3 times faster on average than the conventional methods. And the avergae accuracy of AutoTrain has 2% improved to the

				2	S	08/02/2022 13:13

In this paper, we propose an efficient automatic training
system, AutoTrain, to solve small-scale image classification problems. First, we design sample equalization scheme in data augmentation to improve the performance of training on uneven data. Then, a Bayesian optimization-based strategy controller is introduced to rapidly find the strategy applied in data augmentation. According to the label information, the AutoTrain generates different strategy sub-set for different types of images. It effectively accelerates the strategy optimization process by reducing the search space. Next, we present a dynamic adjustment model to fit tasks with different scales and complexity. We implement a modelselect module to automate model selection and adjustment. Additionally, the transfer learning and early stopping technology are used to accelerate the model training

				3	S	08/02/2022 13:13

AutoAugment is a method proposed by Google's Ekin D.
Cubuk et al [16]. To automatically search for suitable data enhancement strategies. This method creates a search space for data enhancement strategies, and uses a reinforcement learning-based search algorithm to select specific data sets. Appropriate data enhancement strategies. In addition, the data enhancement strategies learned from one data set can be well migrated to other similar data sets. The workflow of AutoAugment is as follows:

				4	S	08/02/2022 13:14

AutoTrain's Framework In response to the shortcomings of DeepAugment men-
tioned in the previous section, we make corresponding improvements and design a more efficient model automated training system for small-scale classification—AutoTrain.

				5	S	08/02/2022 13:14

DeepAugment is an automation tool focused on data augmentation created by Ozmen [15]. Compared with AutoAugment, DeepAugment reduces the error rate of the child model by optimizing its' architecture, the usage of random sampler on validation set solves the problem of overfitting. Instead of using reinforcement study, DeepAugment uses a Bayesian's algorithm to get the best data augmentation strategy, which is faster than AutoAugment's method. Through the above improvements, DeepAugment has 50 times faster training speed compared to AutoAugment. The workflow of AutoAugment is as follows:

				6	S	08/02/2022 13:14

Files\\Task-Specific Automation in Deep Learning Processes

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On
No	Web of science	0.0607	3			
				1	S	03/02/2022 15:12
				2	S	03/02/2022 15:12
				3	S	03/02/2022 15:12

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On

Files\\The Machine Learning Bazaar~ Harnessing the ML Ecosystem for Effective System Development

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On
No	Google Scholar	0.0301	3			
				1	S	03/02/2022 14:38

To address these problems, we introduce the Machine Learning Bazaar, a new framework for developing machine learning and automated machine learning software systems. First, we introduce ML primitives, a unified API and specification for data processing and ML components from different software libraries. Next, we compose primitives into usable ML pipelines, abstracting away glue code, data flow, and data storage. We further pair these pipelines with a hierarchy ofAutoML strategies — Bayesian optimization and bandit learning. We use these components to create a general-purpose, multi-task, end-to-end AutoML system that provides solutions to a variety ofdata modalities (image, text, graph, tabular, relational, etc.) and problem types (classification, regression, anomaly detection, graph matching, etc.). We demonstrate 5 real-world use cases and 2 case studies of our approach

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On
				2	S	03/02/2022 14:42
				3	S	03/02/2022 14:43

Nodes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications\Cloud_based_ML

PDF

Files\\ThunderML~ A Toolkit for Enabling AI~ML Models on Cloud for Industry 4.0

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On
No	Google Scholar	0.0742	8			
				1	S	03/02/2022 11:15

In order to address these issues, we have developed ThunderML, a Python-
based toolkit that makes the creation and deployment of purpose built AI models for industrial applications easier. ThunderML leverages many open source frameworks such as scikit-learn, Tensorflow, and Keras. The extension points are predominantly in terms of how we have built out a series of useful modeling functions and industrial solution templates to expedite the task of building and deploying AI for industrial applications. ThunderML is flexible enough to run on local hardware as well as providing an easier path to using common cloud service provider platforms for enhanced scalability in training and convenient model deployment services. Before we proceed, it's worth briefly giving a few examples of purpose built industrial solution templates available in ThunderML:
– Time Series Prediction (TSPred): Flexible solution for forecasting time series from historical data in industries.
– Failure Pattern Analysis (FPA): Predicting imminent failures for assets using IoT sensor data and past failure history data;
– Root Cause Analysis (RCA): Building interpretable models to assist plant operators track down the root causes for product quality deviances on batch or continuous process lines;
– Anomaly Analysis: Building unsupervised/semi-supervised models to identify anomalous behaviors of manufacturing assets;
– Cognitive Plant Advisor (CPA): Combines advanced AI to build a predictive model of one or more key process outputs such as throughput and yield and uses these models within a business objective optimization problem to suggest optimal process settings to plant operators.
In summary, ThunderML can also help alleviate the skills gap issue that has
hampered AI adoption in many industries. In our experience, technically adept (but not necessarily experts in AI personnel) can use

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On
				2	S	03/02/2022 11:16

Our contribution in this paper is the design and implementation of ThunderML. We elaborately discuss how ThunderML expedites the AI modeling workflow by giving practitioners an easier path for doing advanced modeling work leveraging cloud-based platforms for training and deployment. We then provide a use case to demonstrate the benefits of ThunderML in practice for a very general and widely applicable problem.

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On
				3	S	03/02/2022 11:16
				4	S	03/02/2022 11:16
				5	S	03/02/2022 11:17
				6	S	03/02/2022 11:17
				7	S	03/02/2022 11:17
				8	S	03/02/2022 11:18

Nodes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications\ML System Quality

PDF

Files\\Cats are not fish~ deep learning testing calls for out-of-distribution awareness

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On
No	ACM Digital library	0.0071	1			
				1	S	08/02/2022 12:36

Although recent progress has been made in designing novel testing techniques for DL software, which can detect thousands of errors, the current state-of-the-art DL testing techniques usually do not take the distribution of generated test data into consideration. It is therefore hard to judge whether the ”identified errors” are indeed meaningful errors to the DL application (i.e., due to quality issues of the model) or outliers that cannot be handled by the current model (i.e., due to the lack of training data). Tofill this gap, we take thefi rst step and conduct a large scale empirical study, with a total of 451 experiment configurations, 42 deep neural networks (DNNs) and 1.2 million test data instances, to investigate and characterize the impact of OOD-awareness on DL testing. We further analyze the consequences when DL systems go into production by evaluating the effectiveness of adversarial retraining with distribution-aware errors. The results confirm that introducing data distribution awareness in both testing and enhancement phases outperforms distribution unaware retraining by up

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On

Files\\How Teams Communicate about the Quality of ML Models~ A Case Study at an International Technology Company

No	ACM Digital library	0.0067	1			
				1	S	07/02/2022 16:29

Our interviews and survey focused on observing what quality means to team members working on ML models and how the quality of ML models is communicated within big teams holding different roles. Through our observations, we identified challenges that team members face in perceiving the quality ofmodels and how they tackled. Some teams overcome the challenges in communicating ML models by having a middleman, usually a PM or SE, to communicate model quality aspects between model developers and other team members who are non-ML experts (e.g., UX designers, legal, sales, etc.). This causes some information to be lost in translation. As some of our participants reported, involving ML developers in meetings with other team members has proven to be more efficient and fruitful to the discussion and overall success of a product. In this section, we start by synthesizing and discussing the main challenges in communicating the quality ofML models to a wide audience. After that, we discuss best practices in communicating quality through five lenses (who and with whom, what, form, and goal). Throughout the discussion, we use the word stakeholders to refer to internal employees who are part of the same software organization but are from different teams, such as UX, legal, sales, etc.

Files\\On misbehaviour and fault tolerance in machine learning systems

No	Web of science	0.0272	16			
				1	S	07/02/2022 10:43

As such, in this paper, the goal is to gather additional knowledge on fault tolerance solutions and beyond, and the practical applicability and reasoning of the solutions – which are used and considered useful. We reached out to experienced software architects familiar with ML through their work. In this way, we aim to shed light on which design solutions are seen as useful by experts, which are not, and which need additional studying, thus answering the lack of research on the functionality of deployed ML models identified by Zhang et al. [5].

				2	S	07/02/2022 10:45

ML testing into offline and online testing. Offline testing is basically ML model validation [15], whereas online testing includes the initial testing after model deployment, and the measures taken to ensure correct functionality beyond initial tests, such as monitoring and other fault tolerant patterns. However, the papers yielded by their search presented mostly offline testing, and very little online testing.

				3	S	07/02/2022 10:45

Solution proposals selection The patterns chosen are either mentioned in earlier research
in the context of ML, presented in materials for traditional software, or are a modification of some of these solutions which we

				4	S	07/02/2022 10:45

Fault-tolerance solution proposals Input checker (used by Jonsson et al. [16]) is a component
that aims to prohibit such inputs from entering the ML model that could activate the ML model's faults. Thus, the faults are tolerated by limiting the potential situations in which they could cause errors.

				5	S	07/02/2022 10:45

Output checker (used by Prado et al. [17] and Li et al. [18],
also known as acceptance test [11]) is a component which detects errors by assessing ML model's outputs and prevents errors from propagating further into other parts of the system.

				6	S	07/02/2022 10:46

On misbehaviour ofML systems (RQ1) The mentioned kinds of misbehaviour were unexpected input-output pairs, poor quality of incoming data, and decay of the ML model over time. The first could be considered the simplest kind of erroneous behaviour: with some

				7	S	07/02/2022 10:46

Misbehaviour is usually the result of faulty implementation,
misuse of the model's results, or a poor or buggy model.

				8	S	07/02/2022 10:46

On the role offault tolerance in ML software (RQ2) The need for and role of fault tolerance was deemed to be
contextual and varying. As mentioned earlier in Section

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On
				9	S	07/02/2022 10:47

Patterns used as fault tolerance (RQ3) In this section, we present what the respondents thought about the fault tolerance solutions as presented in the study

				10	S	07/02/2022 10:47

Input checker Input checkers were rarely being used in practice. However,
there is use for input checkers, when certain conditions are met. First of all, hard limits on inputs were seen – at best – as an efficient way to prevent poor quality data from entering the model. For example, broken data or data beneath or above some threshold can be filtered out. It may be that the model cannot handle null values, or its results may be unreliable if the user – for example – has not watched enough videos for a recommendation.

				11	S	07/02/2022 10:48

Input distribution observing Input distribution observing was not one of the original study propositions was but, however, mentioned by every respondent. The statistics of the inputs are measured over time, and deviations in the statistics either alert the developers, or potentially lead to some predefined actions being taken.

				12	S	07/02/2022 10:47

Output checker The respondents considered hard limits on outputs more use-
ful than their counterparts for inputs. Again, business rules or easily confirmable erroneous outputs with direct consequences to users are what set the rules for outputs. For example, business executives might not even approve an autonomous

				13	S	07/02/2022 10:48

Output distribution observing Our study proposal of comparing outputs to historical data
was not triumphant when it concerned single outputs. Instead, monitoring the distribution of outputs in a manner similar to inputs in Section 5.3.2 is something that the respondents mentioned frequently.

				14	S	07/02/2022 10:48

Model observers Measuring the resource consumption of the ML model was
mostly disregarded as a tool for fault tolerance for a ML system, but was considered more as a development tool to indicate non-optimal solutions when building an ML model.

				15	S	07/02/2022 10:48

Redundant models Having multiple divergent models as recovery blocks to hand
the inputs over to was seen as somewhat useful as a fall-over approach in case the main model not give any outputs, or if it was possible to detect erroneous outputs.

				16	S	07/02/2022 10:48

Fall-over options Over the course of the interviews, fall-over procedures were mentioned by the respondents. Essentially, a fall-over means what to do when an error is detected. The recovery blocks of the previous subsection fall into this category as well: when an error is detected, the input is handed over to another model which acts as a fall-over component.

Files\\On testing machine learning programs

No	Web of science	0.0013	1			
				1	S	07/02/2022 10:21

Dead experimental code paths which happens when code is written for rapid prototyping to gain quick turnaround times by performing additional experiments simply by tweaks and experimental code paths within the main production code.

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On

Files\\Quality Assurance for AI-Based Systems~ Overview and Challenges (Introduction to Interactive Session)

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On
No	Scopus	0.0454	3			
				1	S	04/02/2022 14:01

For instance, additional quality properties of AI components and AI-based
systems have to be taken into account. Zhang et al. [5] consider the following quality properties:
– Correctness refers to the probability that an AI component gets things right. – Model relevance measures how well an AI component fits the data. – Robustness refers to the resilience of an AI component towards perturbations. – Security measures the resilience against potential harm, danger or loss made via manipulating or illegally accessing AI components.
– Data privacy refers to the ability of an AI component to preserve private data information.

				2	S	04/02/2022 14:00

				3	S	04/02/2022 14:02

we defined the three dimensions artifact type (i.e., data, model, framework, and system), process (from isolated to continuous), and quality characteristics (with respect to software quality, quality-in-use, and data quality). Furthermore, we elaborated on the key challenges of (1) understandability and interpretability of AI models, (2) lack of specifications and defined requirements, (3) need for validation data and test input generation, (4) defining expected outcomes as test oracles, (5) accuracy and correctness measures, (6) non-functional properties of AI-based systems, (7) self-adaptive and self-learning characteristics, and (8) dynamic and frequently changing environments. In order to properly address the challenges raised in this paper and to enable
high quality AI-based systems, first and foremost, exchange of knowledge and ideas between the SE and the AI community is needed. One channel of exchange is education or training through dedicated courses [29]ormedia[30]. Another one are dedicated venues for exchange and discussion of challenges on quality assurance for AI-based systems

Files\\Quality Management of Machine Learning Systems

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On
No	Scopus	0.0239	5			
				1	S	04/02/2022 13:51

The goal of this paper is to provide an overview of such a framework built upon tools and methodology available today and identify gaps for new software engineering research. The focus of this paper is on AI systems implemented using machine learning. A popular ML technique is the use of Deep Neural Networks (DNN). This paper uses AI and ML interchangeably.

				2	S	04/02/2022 13:52

				3	S	04/02/2022 13:56

3.1 Where are the bugs?

				4	S	04/02/2022 13:56

3.2 Quality Improvement Tasks for ML systems
This section describes the suggested tasks to find defects in the artifacts described in Section 3.1 and resolve them. These are traditional activities modified to reflect the inclusion of the ML component in the application. Due to space limitations, reference to any specific technique or tool is meant to provide an example, rather than an exhaustive list. Quality improvement tasks that address the unique aspects of assessing ‘Trust’ in ML systems are described in Section 3.3.

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On
				5	S	04/02/2022 13:56

3.3 AI Trust Assessment

Nodes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications\MLOps

PDF

Files\\MLOps Challenges in Multi-Organization Setup~ Experiences from Two Real-World Cases

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On
No	Scopus	0.0109	1			
				1	S	07/02/2022 11:27

To improve, we need integration mechanisms for ML/AI, analogous to integration patterns in information systems [10] but applicable at the level of AI/ML features, to create multiorganization AI/ML systems. Like with information systems, there are several challenges that need to be tackled, including integration interfaces, scaling, privacy, governance, and so on. In this paper, we focus on integration and scaling of
systems that include ML components. The setup we assume is that of continuous deployment [6], where new versions of the system can be rapidly deployed – often referred to DevOps [3], [5] in software development. When also ML components are deployed in a similar

Files\\Towards MLOps~ A Framework and Maturity Model

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On
No	Scopus	0.0469	5			
				1	S	03/02/2022 10:15

The adoption of continuous software engineering practices such as DevOps (Development and Operations) in business operations has contributed to significantly shorter software development and deployment cycles. Recently, the term MLOps (Machine Learning Operations) has gained increasing interest as a practice that brings together data scientists and operations teams. However, the adoption of MLOps in practice is still in its infancy and there are few common guidelines on how to effectively integrate it into existing software development practices. In this paper, we conduct a systematic literature review and a grey literature review to derive a framework that identifies the activities involved in the adoption of MLOps and the stages in which companies evolve as they become more mature and advanced. We validate this framework in three case companies and show how they have managed to adopt and integrate MLOps in their large-scale software development companies.

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On
				2	S	03/02/2022 10:25

The use of MLOps enables automation, versioning, reproducibility, etc., with successful collaboration of required skills such as data engineer, data scientist, ML engineer/developer [40] [29]. For example, data scientists must specialize in SE skills such as modularization, testing, versioning, etc. [36]. Supporting processes formalized in policies serve as the basis for governance [31] and can be automated to ensure solution reliability and compliance [31]. MLOps also support explainability (GDPR regulation [25]) and audit trails [40]

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On
				3	S	03/02/2022 10:25

MLOPS FRAMEWORK AND MATURITY MODEL Based on the SLR and the GLR, we derive an MLOps
framework that identifies the activities involved in MLOps adoption. Figure 2 depicts the MLOps framework. The entire framework is divided into three pipelines: a) Data Pipeline b) Modeling Pipeline and c) Release Pipeline.

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On
				4	S	03/02/2022 10:25

Based on the SLR and the GLR, we present a maturity model in which we outline four stages in which companies evolve when adopting MLOps practices. The four stages are a) Automated Data Collection b) Automated Model Deployment c) Semi-automated Model Monitoring and d) Fully-automated Model Monitoring. These stages capture key transition points in the adoption of MLOps in practice. Below, we detail each MLOps stage and preconditions for a company to reach this stage.

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On
				5	S	03/02/2022 10:26

Nodes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications\pipeline jungles

PDF

Files\\On the Co-evolution of ML Pipelines and Source Code - Empirical Study of DVC Projects

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On
No	Web of science	0.0033	1			
				1	S	04/02/2022 22:59

As such, a new breed of data and model versioning tools
have appeared to support data engineers and scientists [3]. Popular tools comprise DVC [4], MLFlow [5], Pachyderm [6], ModelDB [7] and Quilt Data [8]. They typically combine the ability to specify data and/or model pipelines, with advanced versioning support for data/models, and the ability to define and manage model experiments.

Nodes\\Maintainable ML\\Available solutions for maintaining a ML systems\Building ML Systems and applications\Training distributed

PDF

Files\\Bighead~ A Framework-Agnostic, End-to-End Machine Learning Platform

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On
No	IEEE	0.0315	2			
				1	S	08/02/2022 12:56

Many machine learning platforms have been developed at
various companies. We briefly overview some major works in this section. TFX [3] is an end-to-end machine learning platform developed by Google, which spans from prototyping to production. It exclusively supports TensorFlow [7] as the model framework. Kubeflow [8] is also developed at Google, focusing on serving models in Kubernetes. MLflow [4] is developed and open sourced by Databricks. It is integrated with several cloud service providers, such as AWS and Azure. H2O [5] is a open source machine learning platform implemented in JVM with API libraries in several languages. Skymind Intelligence Layer [9], built on top of DeepLearning4J, offers model serving and scalability in its enterprise edition. Several in-house platforms cover many aspects of the machine learning workflow, such as Uber's Michelangelo [6], Facebook's FBLearner Flow [10], and Groupon's Flux [11]. However, these platforms are internal and not yet open sourced. Data Robot [12] is a popular proprietary system that offers features for automated machine learning. Several systems like Polyaxon [13], Comet [14], and Atalaya [15] provides model serving. Cloud service providers offer systems that enable the building, serving, and management of models, including Amazon's SageMaker [16], Microsoft Azure Machine Learning

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On
				2	S	08/02/2022 12:57

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On

Files\\Ultron-AutoML~ an open-source, distributed, scalable framework for efficient hyper-parameter optimization

No	IEEE	0.0061	1			
				1	S	03/02/2022 10:07

The framework supports the creation of datapipelines to stream batches of shuffled and augmented data from a distributed file system. This comes in handy for t raining Deep Learning models based on self-supervised, semi-supervised or representation learning algorithms over large training datasets. We demonstrate the framework's reliability and efficiency by running a BERT pre-training job over a large training corpus using pre-emptible GPU compute targets. Despite the inherent unreliability of the underlying compute nodes, the framework is able to complete such long running jobs at 30% of the cost

Nodes\\Maintainable ML\\Available solutions for maintaining a ML systems\Data Engineering\Data cleaning

PDF

Files\\Data Cleaning for Accurate, Fair, and Robust Models~ A Big Data - AI Integration Approach

No	Scopus	0.0644	6			
				1	S	07/02/2022 23:21

We contend that it is time to extend the notion of data cleaning for modern machine learning needs. We identify dependencies among the data preprocessing techniques and propose MLClean, a unified data cleaning framework that integrates the techniques and helps train accurate and fair models. This work is part of a broader trend of Big data – Artificial Intelligence

				2	S	14/02/2022 14:33

As a running example, suppose we are cleaning a set of
training examples in Table 1 (small for illustration purposes). This data is not clean in the sense that e2 and e3 refer to the same person because their ages are the same and Joe is an abbreviation of Joseph. (In comparison, e4 and e5 are not the same person because they have very different ages.) In addition, e6 has an unusually-high age, which can be viewed as an incorrect value. Hence, cleaning this data may involve merging e2 and e3 to a single example e23 and fixing or removing e6's age.

				3	S	14/02/2022 14:33

More recently, there are mitigation techniques for fixing unfairness, which can be done before (pre-processing), during (in-processing), or after (post-processing) model training [2]. These techniques typically tradeoff some model accuracy in order to improve model fairness. Among them, we focus on the pre-processing approach where the example weights are adjusted (i.e., reweighed [3]) to maximize fairness. For example, a simplified reweighing technique for demographic parity is to increase the weights of positivelylabeled examples in sensitive groups whose ratio ofweighted positive labels is lower than other groups.

				4	S	14/02/2022 14:33

A popular solution is to make the model training more robust. Another approach that is gaining interest is sanitizing the poisoned data before it is used in training. Data poisoning attacks have recently become more sophisticated [9], and

				5	S	14/02/2022 14:29

MLCLEAN
Since data cleaning, unfairness mitigation, and data sanitization are ultimately preprocessing the same dataset, it makes sense to unify them. The naïve approach of applying each technique independently in any sequence can be problematic for several reasons. Simply ignoring the dependencies between preprocessing techniques may result in incorrect results. For example, ifwe reweigh examples and then attempt to remove duplicates, then the reweighing may need to be done again to ensure fairness. Moreover, running one operation at a time may have efficiency issues due to redundant operations on the data.
3.1 Basic Architecture

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On
				6	S	14/02/2022 14:29

Beyond removing duplicates, data cleaning can be any general process like HoloClean [11]. Data sanitization can also employ more sophisticated defenses [9]. Of course, one should carefully analyze the possible interactions between each cleaning and sanitization combination.

Files\\Data collection and quality challenges for deep learning

No	Web of science	0.0031	1			
				1	S	14/02/2022 14:43

We cover the recent CleanML [8] work, which systematically studies the impact of data cleaning on the accuracy of the model trained on that data.

Nodes\\Maintainable ML\\Available solutions for maintaining a ML systems\Data Engineering\Data managment

PDF

Files\\Juneau~ data lake management for Jupyter

No	Web of science	0.1325	5			
				1	S	07/02/2022 15:30

In collaborative settings such as multi-investigator laboratories, data scientists need improved tools to manage not their data records but rather their data sets and data products, to facilitate both provenance tracking and data (and code) reuse within their data lakes and file systems. We demonstrate the Juneau System, which extends computational notebook software (Jupyter Notebook) as an instrumentation and data management point for overseeing and facilitating improved dataset usage, through capabilities for indexing, searching, and recommending "complementary" data sources, previously extracted machine learning features, and additional training data.

				2	S	07/02/2022 15:31

In this demonstration, we present a prototype of JUNEAU system,
which provides these capabilities. Our demonstration illustrates how indexing, searching, and reusing tabular data are supported for tabular, CSV, and relational datasets. JUNEAU addresses scientists' need to search for prior tables (and related code) not merely by keyword, but by querying using an existing table and its provenance, to find other related tables. Within the Jupyter environment, users may select a table (dataframe) and directly search for related tables for different purposes. Motivating use cases. We outline the four use cases for finding related tables.
EXAMPLE 1.1 (AUGMENTING TRAINING DATA). Often,
data is captured in multiple sessions (perhaps by multiple users) using the same sensor device or tool. Given a table from one such session, the user may wish to augment his or her data, to form a bigger training or validation set for a machine learning algorithm.
EXAMPLE 1.2 (LINKING DATA VIA ONTOLOGIES). Particul-
arly in the life sciences, records in one database may have identifiers (e.g., "accession numbers") linking to entries in another database or ontology. Such entries may transitively reference other entries, and each brings in additional fields that may be useful. It

				3	S	07/02/2022 15:31

EXAMPLE 1.3 (AUGMENTING FEATURES). Another commo-
n task for data scientists is to find additional or alternative features for the given data instances that may lead to a better performance. Especially in the collaborative setting, one data scientist may perform a specific feature engineering on a data set, while another may do it in a different way. It can be helpful for data scientists to be recommended with other feature engineering possibilities.
EXAMPLE 1.4 (FINDING WORKFLOWS FOR DATA). Given a
widely used and related table, a data scientist may want to see examples of how the table is loaded or cleaned, what analysis have been performed on it, and so on. Generally, this requires us to search for workflows using the table or related tables, potentially featuring specific operations.

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On
				4	S	07/02/2022 15:32

The JUNEAU System replaces Jupyter Notebook's back-end and
extends it user interface. Our back-end "data lake management" subsystem integrates relational and key-value stores to capture and index (1) any external files loaded by the notebooks; (2) intermediate data products produced by computational steps (cells) within the notebooks; (3) versioned cell content and notebook content, as in the right-hand side of Figure 1; (4) indices for rapidly retrieving tables and their provenance. We illustrate the basic architecture and functionality in Figure 4.
As in the existing Jupyter Notebook software, the notebook interface interacts with a kernel (language interpreter) every time the user executes a cell. The cell contents are executed in the kernel, thus updating state in the kernel as well. JUNEAU fetches any new or changed tables (dataframes) from the kernel after each step, and it imports and indexes those in the backend. The user may interactively select any table within the notebook,
and query the JUNEAU search engine for other tables already stored and indexed in the data lake which are related to the selected item. As we described in the introduction, users often want to search for other related tables using an existing table as a model, and possibly adding other filter criteria such as author, attribute name or content, or the name of a computational process that was involved in the provenance of the search result. The search may not purely be based on whether other tables have a common schema or joinable fields, but may also consider similarity of computational (provenance) steps.

				5	S	07/02/2022 15:33

Files\\Shuffler~ A Large Scale Data Management Tool for Machine Learning in Computer Vision

No	Scopus	0.1040	6			
				1	S	04/02/2022 13:07

In this work, we present Shuffler, an open source tool that makes it easy to manage large computer vision datasets. It stores annotations in a relational, human-readable database. Shuffler defines over 40 data handling operations with annotations that are commonly useful in supervised learning applied to computer vision and supports some of the most well-known computer vision datasets. Finally, it is easily extensible, making the addition of new operations and datasets a task that is fast and easy to accomplish.

				2	S	04/02/2022 13:03
				3	S	04/02/2022 13:10

In this work, we close this gap by proposing a software tool-
box, Shuffler, designed specifically for manipulating annotations. It employs widely known relational databases and the associated SQL query language for storing and manipulating annotations. The proposed toolbox is heavily based on SQL and allows to chain multiple operations in a single command. Annotations are stored in an relational database (Sqlite, MySql, ...) with schema designed to cover the bulk of the common tasks in computer vision. The proposed solution satisfies the following properties: • it has basic manipulation tools and allows to easily add new functions;
• annotations are fast to load and to modify and convenient to store;
• annotations are stored in a human readable format that can be manually edited;
• it is agnostic to the format of how images are stored on disk; • it supports image-level classification, object detection, semantic segmentation, and object matching tasks in computer vision.

				4	S	14/02/2022 15:49

First, some systems are designed specifically for annotating data
for Computer Vision applications. Examples include publicly available LabelMe [17], VGG Image Annotator [7], and CVAT3, as well

				5	S	14/02/2022 15:51

as commercial Supervisely4, Playmate5, and Labelbox6. These systems offer sophisticated tools for human annotators to label images in order to prepare training data for different types of Machine Learning tasks. Though our proposed toolbox, Shuffler, offers basic functionality for image labelling, its primary focus is processing the output of such image annotation systems. Second, an important part of a Machine Learning pipeline is
loading and augmenting image data. Numerous libraries, including a library from NVIDIA, DALI7, have been proposed for this task. In Figure 2, we refer to this part of the pipeline as step 3. In turn, Shuffler is employed on step 2 to prepare a dataset of training data that will be further loaded and augmented during training. Next, end-to-end product life-cycle management systems have
been proposed, such as ModelHub [13], and commercial Allegro8. These systems focus on Machine Learning model management, while the goal of Shuffler is to provide instruments to manage training data.

	Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On
					6	S	04/02/2022 13:14

Nodes\\Maintainable ML\\Available solutions for maintaining a ML systems\Data Engineering\Data Pipeline

PDF

Files\\On testing machine learning programs

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On
No	Web of science	0.0050	1			
				1	S	07/02/2022 10:22

Conceptual issues. One key assumption behind the training process of supervised ML models is that the training dataset, the validation dataset, and the testing dataset, which are sampled from manually labeled data, are representative samples of the underlying problem. Following the concept of Empirical Risk Minimization (ERM), the optimizer allows finding the fitted model that minimizes the empirical risk; which is the loss computed over the training data assuming that it is a representative sample of the target distribution. The empirical risk can correctly approximates the true risk only if the training data distribution is a good approximation of the true data distribution (which is often out of reach in real-world scenarios). The size of the training dataset has an impact on the approximation goodness of the

Files\\On the Co-evolution of ML Pipelines and Source Code - Empirical Study of DVC Projects

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On
No	Web of science	0.0033	1			
				1	S	04/02/2022 22:59

As such, a new breed of data and model versioning tools
have appeared to support data engineers and scientists [3]. Popular tools comprise DVC [4], MLFlow [5], Pachyderm [6], ModelDB [7] and Quilt Data [8]. They typically combine the ability to specify data and/or model pipelines, with advanced versioning support for data/models, and the ability to define and manage model experiments.

Files\\Ultron-AutoML~ an open-source, distributed, scalable framework for efficient hyper-parameter optimization

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On
No	IEEE	0.0061	1			
				1	S	03/02/2022 10:06

The framework supports the creation of datapipelines to stream batches of shuffled and augmented data from a distributed file system. This comes in handy for t raining Deep Learning models based on self-supervised, semi-supervised or representation learning algorithms over large training datasets. We demonstrate the framework's reliability and efficiency by running a BERT pre-training job over a large training corpus using pre-emptible GPU compute targets. Despite the inherent unreliability of the underlying compute nodes, the framework is able to complete such long running jobs at 30% of the cost

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On

Nodes\\Maintainable ML\\Available solutions for maintaining a ML systems\Data Engineering\Data preprocessing

PDF

Files\\A hybrid method for missing value imputation

No	ACM Digital library	0.0568	5			
				1	S	11/02/2022 13:00

A widely used imputation method that can be found in libraries of the most noted statistical and Machine Learning suites is IRMI. In this work, we propose a variant of IRMI in order to maintain the advantages of this famous imputation method, while outperforming its traditional variant used in many Machine Learning software tools. To achieve this, the benefits of boosting as well as decision tree theory are exploiting. To test the efficiency of our method, a series of experiments over 30 datasets was executed, measuring the classification accuracy of the proposed method to prove that outperforms its rivals, which include classic, as well as more sophisticated imputation strategies. Finally, the results of our study are provided, along with the conclusions that arise from them.

				2	S	11/02/2022 13:03

The purpose of this work is to propose an efficient imputation method for missing values based on the well-known IRMI imputation strategy, which stands for Iterative Robust Modelbased Imputation. As implies its name, the idea behind this algorithm is quite simple. To provide an estimation of a missing data value, IRMI uses the missing value as a target value and the remaining variables as regressors. This way, the entire dataset is used as a multivariate model whose final prediction is the computation of an estimation of the initial missing value. IRMI was initially introduced and described in [28] as an improvement of IVEWARE algorithm [23], while here is provided only a brief description of its function. The first step of the IRMI algorithm consists of an
initialization of all missing values in the dataset, by using a simple imputation method, like k-nearest neighbors. Then the variables are sorted concerning the initial number of missing values existing in them. After that, a two-step iterative procedure takes place consisting of two nested loops. In every iteration of the inner loop, the missing values of one variable are updated, starting with the one that contains the greater amount of missing information, and moving towards the one which contains the less. In order to achieve this, the cells of the considering variable are split into two subsets: one that contains the cells with initially unknown values and a second, which contains the

				3	S	11/02/2022 13:03

In the presentation of IRMI direction guidelines are given
about methods that can be used according to the type of the target variable. Considering continuous and categorical values, in classic IRMI a selection of a robust regression model (Logistic algorithm is preferred usually in IRMS's software implementations) and Linear Regression are suggested respectively. Our proposed method differentiates in this part, using:
 M5P regression trees [22] for imputing numeric values.
a boosting learner, Logitboost [10] [25], in charge of imputing categorical values, and

				4	S	11/02/2022 13:03

				5	S	11/02/2022 13:04

Learners that are not statistically independent are connected with bold horizontal lines, while statistically independent learners are not connected. In all six cases, the proposed method was statistically independent, as can easily be assumed by the provided CD plots. According to the results provided by computing the accuracy metric and the statistical test that followed, it is almost safe to conclude that our method outperforms its rivals in all six scenarios, not only by achieving high accuracy but also proving its statistical independence

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On

Files\\An Empirical Study of the Impact of Data Splitting Decisions on the Performance of AIOps Solutions

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On
No	Google Scholar	0.0451	16			
				1	S	10/02/2022 11:00

The contributions of this article are:
(1) This is the first work that assesses the performance impact ofvarious data splitting decisions on AIOps solutions. The findings and the proposed techniques in this article can be useful to machine learning engineers or software engineering researchers interested in improving the quality and maintainability of AIOps solutions.
(2) Our results show that problems such as data leakage (caused by decisions during model training and evaluation) and concept drift (caused by the evolution of data) can easily appear in AIOps solutions ifnot being careful while deciding on various data splitting strategies. Such problems may severely impact the performance ofAIOps solutions while being deployed in the field.
(3) To mitigate the risks of the various problems arising from data splitting decisions, we also proposed suggested techniques and demonstrated their effectiveness in our case studies. In particular, we observe that using a time-based splitting of training and validation datasets can reduce data leakage and provide a more reliable evaluation. We also observe that periodically updating AIOps models can help mitigate the impact of concept drift, while the frequency of model updating should be cautiously considered.

				2	S	15/02/2022 11:43

Prior work proposed many AIOps solutions to address various problems in the operations of largescale software and systems, such as incident prediction [5, 16, 24, 49, 51, 57, 70, 89], anomaly detection [31, 50], ticket management [90, 91], issue diagnosis [55], and self healing [18, 19, 52, 53]. For example, Lin et al. [51]and Li et al.[49] leverage temporal data (e.g., CPU and memory utilization metrics, alerts), spatial data (e.g., rack locations), and config data (e.g., memory size) to predict node failures in large-scale cloud computing platforms. El-Sayed et al. [24] and Rosa et al. [70] learned from the trace data to predict job failures in the Google cloud computing platform. Botezatu et al. [5], Mahdisoltani et al. [57], andXuetal. [89] leveraged disk-level sensor data and systemlevel events to predict disk failures in operations of large-scale cloud platforms. As illustrated in Figure 1, ML modeling, in particular, supervised learning, in the context of AIOps usually faces three data splitting-related challenges: the imbalanced data challenge in model training, the data leakage challenge in model training and evaluation, and the concept drift challenge in model maintenance. Table 1 and Table 2 list prior AIOps work that leverages supervised learning and unsupervised learning techniques, respectively. For the works using supervised learning, we summarize how they handle the three challenges in different ML modeling phases. Below, we discuss prior AIOps solutions that rely

				3	S	10/02/2022 11:02

Handling imbalanced data. Operation data is often very imbalanced [5, 24, 49, 51, 57].
Therefore, AIOps solutions usually apply data rebalancing techniques (e.g., over-sampling, undersampling, SMOTE, ROSE) to make the modeled classes more balanced and produce more accurate models [44, 80]. For example, Botezatu et al. [5] and Mahdisoltani et al. [57] use under-sampling approaches (i.e., randomly reducing the samples of the majority class) to balance the samples of failed disks and normal disks in their tasks of disk failure prediction. El-Sayed et al. [24]use an over-sampling approach (i.e., making random duplication ofthe minority class) to balance the samples offailed jobs and normally terminated jobs in their tasks ofjob failure prediction. Xu et al. [89] and Chen et al. [16] use the SMOTE over-sampling approach [13] to balance their classes in the tasks of disk failure prediction and service outage prediction, respectively. This work does not explore the impact of data rebalancing techniques on AIOps solutions, as data rebalancing has been extensively discussed in prior work (e.g., References [49, 51, 80]). Instead, we use an under-sampling approach to balance the classes in both our studied datasets, as done in Botezatu et al. [5] and Mahdisoltani et al. [57].

				4	S	10/02/2022 11:03

Handling data leakage. Prior studies [5, 24, 57, 66] usually randomly split the dataset into a
training set and a validation set. For example, El-Sayed et al. [24] randomly split the whole Google cluster trace dataset [88] into 70% training data and 30% validation data. Botezatu et al. [5]and Mahdisoltani et al. [57] randomly split the Backblaze disk stats dataset into 80% training data and 20% validation data, and 75% training data and 25% validation data, respectively. In comparison, some prior studies use a time-based approach to split training and validation data, which ensures that the training data always occurs before the validation data [49, 51, 71, 89]. In this work, we analyze the existence of data leakage in the studied operation datasets (RQ1).
Then, we evaluate the impact of using a time-based splitting (i.e., considering the temporal order in the data) instead of random splitting on model evaluation (RQ2).

				5	S	10/02/2022 11:04

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On
				6	S	10/02/2022 11:05

The different model evaluation scenarios are illustrated in Figure 3 and detailed below. • Baseline splitting:We use a baseline splitting strategy that trains a model using all the past data before predicting each data sample, which intuitively should yield better performance than other valid splitting strategies. Prior work [48] uses a similar approach to evaluate the performance of predicting log changes. Specifically, we first randomly choose N samples as the testing data. For each testing sample, we build a model with all available samples (i.e., samples that have finished before the testing sample) and test the model on the current testing sample. We then combine the prediction results of all the testing samples to calculate the performance. For the Google cluster trace data, we set N as 15,000; for the Backblaze disk stats data, we set N as 150,000. We choose these values to ensure that we have enough samples from the minority classes.

• Random splitting: In this scenario, we first randomly split the data into a training set and a validation set, then we train a model using the training set and evaluate the model on the validation set. We consider five training/validation split ratios that range from 50%/50% to 90%/10%.

• Time-based splitting: In this scenario, we split the data into the training set and testing set based on the temporal order, then we train a model using the training set and evaluate the model on the validation set. We also consider the five training/validation split ratios that

				7	S	10/02/2022 11:05

Data leakage could exist in AIOps solutions that use a random splitting of training and validation datasets, as random splitting achieves a higher model performance than the baseline splitting strategy that leverages all the available past data. We observe that, overall, models that are trained and evaluated on a random splitting have a higher performance than the baseline splitting, which indicates that the random splitting could lead to over-estimation ofmodel performance than the baseline splitting

				8	S	10/02/2022 11:05

Random splitting of training and validation datasets has higher performance than time-based splitting. We observe

				9	S	15/02/2022 11:41

Randomly splitting operation data for the training and validation of a model may cause data leakage problems in AIOps solutions that impact a model’s realistic evaluation.

				10	S	10/02/2022 11:07

The time-based splitting strategy shows a more consistent performance between the validation and the unseen testing datasets compared to random splitting. Figure 7 and Figure 8 show the performance of the models that are trained and evaluated using different data splitting strategies and splitting ratios. Under a random splitting, the evaluatedmodel performance (i.e., on the validation dataset) is less consistent with the performance of the same model on the unseen testing dataset;

				11	S	10/02/2022 11:06
				12	S	10/02/2022 11:07

The time-based splitting strategy provides a more realistic evaluation of an AIOps
model when it is retrained on all the available data and applied to future unseen data. The estimated performance of a model (obtained on the validation dataset) is closer to the

				13	S	10/02/2022 11:07

While prior work relies on the random splitting of training and validation sets, their reported performance on the validation set could be higher than on the unseen testing data, which is a biased evaluation. The bias is particularly larger when specific models (e.g., CART) or a very large splitting ratio (e.g., 90%/10%) is used. On the contrary, the timebased splitting is more appropriate for AIOps model evaluation, since it produces more consistent performance between the validation and unseen testing data.

				14	S	10/02/2022 11:07

Concept drift exists in the operation data. Figure 12 describes the concept drift in different
time periods of the studied datasets. We observe that many of the time periods show a concept drift from its previous period. For example, on the Google dataset, the RF, NN, and CART models indicate that 70% (19 out of 27) time periods exhibit concept drift, and the CART and SVM indicate 18 and 17 periods with concept drift, respectively. On the Backblaze dataset, the CART model shows that all 11 time periods have concept drift from the previous time periods, while the other four models (i.e., RF, NN, RGF, SVM

				15	S	10/02/2022 11:07

Concept drift exists in the operation data, which can be explained by the fact that the relationship between the variables in the operation data evolves over time. Practitioners and researchers should proactively detect and address the problem ofconcept drift in their AIOps

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On
				16	S	10/02/2022 11:08

Due to the existence of concept drift, AIOps models should be updated periodically, as periodically updated models outperform stationary models. In general, increasing the frequency of updating AIOps models can lead to better performance while increasing the modeling cost. However, the performance benefit and modeling cost of increasing the update frequency show very different trends across models and datasets.

Files\\High Performance Data Engineering Everywhere

No	IEEE	0.1055	8			
				1	S	07/02/2022 16:39

We discuss Cylon's architecture in detail, and reveal how it can be imported as a library to existing applications or operate as a standalone framework. Initial experiments show that Cylon enhances popular tools such as Apache Spark and Dask with major performance improvements for key operations and better component linkages. Finally, we show how its design enables Cylon to be used cross-platform with minimum overhead, which includes popular AI tools such as PyTorch, Tensorflow, and Jupyter notebooks.

				2	S	07/02/2022 16:39

One important question is whether those existing Big Data
frameworks utilize the full potential of the computing power and parallelism available to process data. Both Big Data and AI/ML applications spend a goodly amount of time preprocessing data. Minimizing the pre-processing time clearly increases the throughput of these applications. Productivity is another important aspect of such frameworks. Most available data analytics tools are implemented using a rapid programming language such as Java, Python or R. This allows data engineers to develop applications without diverging into the details of complex distributed data processing algorithms. Still, we rarely see these two aspects (high performance and productivity) meet each other in the existing Big Data frameworks [5]. We have also seen the world increasingly moving towards user-friendly frameworks such as NumPy [6], Python Pandas [7] or Dask [8]. Big Data frameworks have been trying to match this by providing similar APIs (for example, PySpark, Dask-Distributed). But this comes at the cost of performance owing to the overheads that arise from

				3	S	07/02/2022 16:40

We believe that a data processing framework focused on
high performance and productivity would provide a more robust and efficient data engineering pipeline. In this paper we introduce Cylon: a high-performance, MPI (Message Passing Interface)-based distributed memory data parallel library for processing structured data. Cylon implements a set of relational operators to process data. While ”Core Cylon” is implemented using system level C/C++, multiple language interfaces (Python and Java (R in future)) are provided to seamlessly integrate with existing applications, enabling both data and AI/ML engineers to invoke data processing operators in a familiar programming language. Large-scale ETL operations most commonly involve map-
ping data to distributed relations and applying queries on them. There are distributed table APIs implemented on top of Big Data

				4	S	07/02/2022 16:40

Cylon1 is a data engineering toolkit designed to work with
AI/ML systems and integrate with data processing systems. This vision is highlighted in Figure 1 where Cylon is shown to support common data structures and systems. It can be deployed either as a library or a framework. Big Data systems like Apache Spark, Apache Flink and Twister2 [2] can use Cylon to boost the performance in the ETL pipeline. For AI/ML systems like PyTorch [12], Tensorflow [13] and MXNet [14], it acts as a library to enhance ETL performance. Additionally, Cylon is being expanded to perform as a generic framework for supporting ETL and efficient distributed modeling of AI/ML workloads. Cylon currently provides a set of distributed data-parallel operators to extract, transform and load structured relational data. These operators are exposed as APIs in multiple programming languages (e.g., C++, Python, Java) that are commonly used in Machine Learning and Artificial Intelligence platforms, enabling tight integration with them. When an operator is invoked in any of these platforms, that invocation is delegated to the ”Core Cylon” framework, which implements the actual logic to perform the operation in a distributed setting.

				5	S	07/02/2022 16:41

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On
				6	S	07/02/2022 16:41

One of the main goals of Cylon is to be a versatile library
which facilitates data processing as a function (DPAF) and thus provide efficient data engineering across different systems. When working over multiple systems, data representation and conversion is a key factor affecting performance and interoperability. Cylon internally uses Apache Arrow data structure, which is supported by many other frameworks such as Apache Spark, TensorFlow, and PyTorch. Apache Arrow can be converted into other popular data structures such as NumPy and Pandas efficiently. In addition our core data structures can work with zero copy across languages. For example, when Cylon creates a table in CPP, it is available to the Python or Java interface without need for data copying. Cylon C++ kernels efficiently support data loading and data processing. These functions can be used either in distributed or local setting. Most of the deep learning libraries like PyTorch, Tensorflow and MXNet are designed on top of such high performance kernels. Cylon APIs are made available to the user in a similar manner. Such designs lead to lower frictions in system integration. With these design principles, we envision the following scenarios where Cylon could work with other systems to create rich

				7	S	07/02/2022 16:41

				8	S	07/02/2022 16:42

C. Cylon, Spark vs. Dask Figure 9 (a) shows a strong scaling wall-clock time com-
parison between Cylon, Spark and Dask. The same strong scaling setup for Inner-Joins was used in this comparison. When comparing with Dask and Spark, Cylon performs better than them on the wall-clock time. For this 200 million line join, it scales better than both of the other frameworks. It should be noted that Dask failed to complete for the world sizes 1 and 2, even when doubling the resources. It continued to fail even with the factory LocalCluster settings, with higher memory. Cylon shows better strong scaling, reaching a higher in-dividual speedup. As shown in Table II for a single worker (serial) Inner-joins, Cylon Hash, Cylon Sort, and Spark took 141s, 164s and 587s respectively. For Union, Cylon and Spark took 34s and 75s respectively. Thus not only does Cylon show better scaling, it achieves a superior wall-clock speed up because its serial case wall-clock time is an improvement on Spark. Figure 9 (b) shows the results for the Union (Distinct) operation. Unfortunately Dask (as of its latest release) does not have a direct API for distributed Union operation. As a result the comparison is limited to Spark and Cylon. As the graph depicts, Cylon performs better than Spark, with more than 2x better

Files\\Inspector gadget~ a data programming-based labeling system for industrial images

No	Web of science	0.0611	7			
				1	S	07/02/2022 15:43

In this work, we expand the horizon of data programming by directly applying it to images without this conversion, which is a common scenario for industrial applications. We propose Inspector Gadget, an image labeling system that combines crowdsourcing, data augmentation, and data programming to produce weak labels at scale for image classification. We perform experiments on real industrial image datasets and show that Inspector Gadget obtains better performance than other weak-labeling techniques: Snuba, GOGGLES, and self-learning baselines using convolutional neural networks (CNNs) without pre-training.

				2	S	07/02/2022 15:44

A conventional solution is to collect enough labels manually
and train say a convolutional neural network on the training data. However, fully relying on crowdsourcing for image labeling can be too expensive. In our application, we have heard of domain experts demanding six-figure salaries, which makes it infeasible to simply ask them to label images. In addition, relying on general crowdsourcing platforms like Amazon Mechanical Turk may not guarantee high-enough labeling quality.

				3	S	07/02/2022 15:45

So far, data programming has been shown to be effective in
finding various relationships in text and structured data [29]. Data programming has also been successfully applied to images where they are usually converted to structured data beforehand [41, 43]. However, this conversion limits the applicability of data programming. As an alternative approach, GOGGLES [9] demonstrates that, on images, automatic approaches using pre-trained models may be more effective. Here the idea is to extract semantic prototypes of images using the pre-trained model and then cluster and label the images using the prototypes. However, GOGGLES also has limitations (see Section 6.2), and it is not clear if it is the only solution for generating training data for image classification.

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On
				4	S	07/02/2022 15:45

We thus propose Inspector Gadget, which opens up a new class
of problems for data programming by enabling direct image labeling at scale without the need to convert to structured data using a combination of crowdsourcing, data augmentation, and data programming techniques. Inspector Gadget provides a crowdsourcing workflow where workers identify patterns that indicate defects. Here we make the tasks easy enough for non-experts to contribute. These patterns are augmented using general adversarial networks (GANs) [13] and policies [7]. Each pattern effectively becomes a labeling function by being matched with other images. The similarities are then used as features to train a multi-layer perceptron (MLP), which generates weak labels. In our experiments, Inspector Gadget performs better overall
than state-of-the-art methods: Snuba, GOGGLES, and self-learning baselines that use CNNs (VGG-19 [36] and MobileNetV2 [33]) without pre-training. We release our code as a community resource [1]. In the rest of the paper, we present the following:
• The architecture of Inspector Gadget (Section 2). • The component details of Inspector Gadget: • Crowdsourcing workflow for helping workers identify patterns (Section 3).
• Pattern augmenter for expanding the patterns using GANs and policies (Section 4).
• Feature generator and labeler for generating similarity features and producing weak labels (Section 5).
• Experimental results where Inspector Gadget outperforms other image labeling techniques – Snuba, GOGGLES, and self-learning baselines using CNNs – where there are few or no labels to start with (Section 6).

				5	S	07/02/2022 15:45
				6	S	07/02/2022 15:45
				7	S	07/02/2022 15:46

Nodes\\Maintainable ML\\Available solutions for maintaining a ML systems\Data Engineering\Data validation

PDF

Files\\On the experiences of adopting automated data validation in an industrial machine learning project

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On
No	ACM Digital library	0.0676	11			
				1	S	04/02/2022 22:34

While several problems in existing data validation tools can
be identified, including implementation errors and decoupling from data cleaning capabilities [13], much focus is on the implementations of these different tools [2, 11, 3, 5]. There is limited reporting on the experiences of adopting the data validation process. The experiences are especially useful for teams that are in the early stages of deploying to production ML-enabled software systems. Adopting the data validation process and tool demands huge engineering resources for development and maintenance [5]. Furthermore, there are no well-established guidelines for establishing a data validation

				2	S	04/02/2022 22:36

Deequ is a tool developed by Amazon Research for automating data quality verification. Deequ allows its users to define ‘unit tests’ for data and combines common quality constraints with user-defined validation code [3]. To perform data validation, the tool relies on declarative user-defined checks on the dataset, for example, isComplete and isUnique checks. The declarative user-defined checks are converted into computations of metrics on data, e.g. different statistical analysis, that can be used to evaluate constraints. After executing data quality verification, the tool reports constraints that succeeded and failed, including information of the computed metric. Although Deequ provides overall data quality report, the tool does not fetch individual records that did not succeed the validations. At Google, the TensorFlow data validation tool [2] is used
to validate trillions of training and serving examples per day. To perform data validation, the tool relies on a data schema

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On
				3	S	04/02/2022 22:37

Finally, Data Sentinel [5] is a data validation platform
developed at LinkedIn. To perform data validation, users use a well-structured configuration file to specify data checks that are desired for specific features. This simplifies the need to write and maintain data checking code. For a given dataset, Data Sentinel computes statistical summaries of the specified features and evaluates the assertions. Eventually, the summaries and validation results are recorded into a dataset profile and validation report. Overall, studies do not provide experiences of adopting
a data validation process and tool by development different teams. The tools presented are also developed by dedicated teams in large companies with several years of experience in deploying to production several ML projects. The few studies that share experiences show slow and poor early adoption with several development iterations [5]. For companies that are in the early stages of deploying ML components to production and from the embedded domain, learning from these experiences is important to help systematize the adoption with minimum resources. This is because the data validation process and tools consume huge amounts of engineering resources

				4	S	04/02/2022 22:37
				5	S	04/02/2022 22:38

IV. RESULTS This section discusses experiences of adopting a data
validation process and tool for ML projects by data science teams at a large software-intensive company in telecommunication domain. The experiences are shaped in form of Best Practices, Benefits and Barriers of adopting data validation in ML projects.
A. Best Practices (RQ1)
Best practices of adopting a data validation process and tool for ML projects are classified in three groups: 1) defining data quality tests, 2) providing actionable feedback, and 3) treating data errors with similar rigor as code.
Results

				6	S	04/02/2022 22:38

1) Defining data quality tests: a data validation process for
ML projects requires having an overview of the level of data (feature, dataset, cross-dataset, data stream) at which

				7	S	04/02/2022 22:39

2) Providing actionable feedback: communicating the out-
put results of data validation in terms of warnings and validation report requires a careful design decision of what and

				8	S	04/02/2022 22:39

3) Treating data errors with similar rigor as code: similar
to software bugs, data errors should be documented, tracked and resolved. Like software unit tests that try to test atomic components in codebase, data validation tests allow designers to quantify the performance of ML models adhering to some specific properties found in ML training datasets. Therefore, structuring data validation tests around the properties of data that the ML model expects to acquire serves as one such approach. In our study, collaborative work, which is important in ML projects,

				9	S	04/02/2022 22:39

B. Benefits (RQ2) The benefits of adopting data validation process and tool
for ML projects include: 1) minimization of manual effort in data preparation; 2) early identification of data errors; 3) a testing approach to ML enabled software systems.

				10	S	04/02/2022 22:40

V. DATA VALIDATION FRAMEWORK (DVF) Based on the experiences, we propose a data validation
framework (DVF) shown in Figure 2 that systematizes the adoption of data validation in ML projects. We group important aspects in the DVF into: A) validation process, B) validation artefacts, C) data validation types, D) data validation tool setup, and E) feedback and mitigation strategy.

				11	S	04/02/2022 22:40

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On

Nodes\\Maintainable ML\\Available solutions for maintaining a ML systems\Data Engineering\Data versioning

PDF

Files\\On the Co-evolution of ML Pipelines and Source Code - Empirical Study of DVC Projects

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On
No	Web of science	0.0122	7			
				1	S	04/02/2022 23:00

As such, a new breed of data and model versioning tools
have appeared to support data engineers and scientists [3]. Popular tools comprise DVC [4], MLFlow [5], Pachyderm [6], ModelDB [7] and Quilt Data [8]. They typically combine the ability to specify data and/or model pipelines, with advanced versioning support for data/models, and the ability to define and manage model experiments.

				2	S	04/02/2022 23:04

~

Despite ML versioning being a young practice in open source repositories, 71.4% of the studied projects use at least two ofthe main DVCfeatures, i.e., data versioning and pipelines. More than half of the DVC files within projects past the experimentation stage are frequently changed, suggesting non-negligible maintenance effort for practitioners.

				3	S	04/02/2022 23:04

Although there is low commit-level coupling amongst the DVC files ofa project, most coupling observed with dvcutilities and software artifacts are automated by DVC. On the contrary, DVC files and software artifacts such as tests and data files are rarely changed together at the commit-level.

				4	S	04/02/2022 23:05

Coupling between DVC and software artifacts are much stronger than would be expected by chance, with one out of four PRs changing source code, and one out of two PRs changing tests, requiring changes to pipeline files.

				5	S	04/02/2022 23:05

VII. Implications of our findings
Implications to ML application developers.

				6	S	04/02/2022 23:06

Implications to ML versioning tool developers/companies.

				7	S	04/02/2022 23:06

Implications to Researchers.

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On

Nodes\\Maintainable ML\\Available solutions for maintaining a ML systems\Data Engineering\Dataset creation

PDF

Files\\Achiever or explorer~ gamifying the creation process of training data for machine learning

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On
No	ACM Digital library	0.0336	6			
				1	S	10/02/2022 12:05

The development ofartiﬁcial intelligence, e. g., for Computer Vision, through supervised learning requires the input of large amounts of annotated or labeled data objects as training data. The creation of high-quality training data is usually done manually which can be repetitive and tiring. Gamiﬁcation, the use of game elements in a non-game context, is one method to make tedious tasks more interesting. This paper proposes a multi-step process for gamifying the manual creation of training data for machine learning purposes. We choose a user-adapted approach based on the results ofa preceding user study with the target group (employees of an AI software development company) which helped us to identify annotation use cases and the users' player characteristics. The resulting concept includes levels of increasing diﬃculty, tutorials, progress indicators and a narrative built around a robot character which at the same time

				2	S	10/02/2022 12:06

The creation of necessary labels is usually performed with the aid of humans. Due to the necessary amount of training data the creation process is typically highly repetitive and quickly turns into a rather unexciting, demotivating task for the annotator.

				3	S	10/02/2022 12:06

for certain psychological outcomes such as motivation, enjoyment, and ﬂow. Previous research shows that a gamiﬁed environment for data annotation has the potential to increase user engagement and gratiﬁcation [12]. Improved user experience is a goal ofgamiﬁcation, as are increased participation, the attraction of a younger audience, optimization ofworkﬂows and increased engagement of users, as well as immediate feedback for the users on their performance [23]. Gamiﬁcation of company workplaces has just recently gained in importance – not only for the training but also to encourage employees in their daily work routine. A tool with well-designed game elements at the workplace can keep employees motivated to perform their tasks [16]. This paper presents the results ofour work aiming at integrating game elements into an existing annotation tool for the creation of training data at the AI product company AI4BD 1. We describe our multi-step development process, thereby laying the foundation for future user studies to investigate the eﬀect of the

				4	S	10/02/2022 12:07

Gamification in video labeling. A game for video annotation was designed in [21]. They thought out three diﬀerent game approaches: a label vote game, an entity annotation where users were asked to assign a certain category to a video segment, a click game, where users had to locate a certain object inside the video and click on it, and a bounding box game, which asked users to draw a box around a speciﬁc object. The last one was implemented and evaluated with the aid of 20

				5	S	10/02/2022 12:07

2.2.2 Tags You Don't Forget: GamifiedTagging ofPersonal Images. Another approach was created by [20] whose scope was the creation of a game, used to annotate personal photos. Two mobile applications were developed (one single, one multiplayer) and evaluated as well as compared to a simple tagging app without any gamiﬁcation.

				6	S	10/02/2022 12:07

2.2.3 Crowdsourcing. Lastly, we analyzed crowdsourcing tools, which often include game elements to engage users. Google Crowdsource [11] is a desktop platform as well as a mobile app, which makes use of humans to improve Google tools such as Google Photos or Google Translate and can be used by anyone who has a Google account.

Files\\Towards Building Robust DNN Applications~ An Industrial Case Study of Evolutionary Data Augmentation

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On
No	IEEE	0.0818	5			
				1	S	11/02/2022 14:40

We evaluate data augmentation techniques in image classification and object detection tasks using an industrial in-house graphical user interface dataset. As the results indicate, the genetic algorithm-based data augmentation technique outperforms two random-based methods in terms of the robustness of the image classification model. In addition, through this evaluation and interviews with the developers, we learned following two lessons: data augmentation techniques should (1) maintain the training speed to avoid slowing the development and (2) include extensibility for a variety of tasks.

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On
				2	S	11/02/2022 14:40

The Worst ofk approach chooses a transformation having the largest value for its loss function among k randomly selected transformations in the training loop. Gao et al. proposed a search-based method called Sensei, which finds an effective transformation with a genetic algorithm [4]. Their evaluation reported that Sensei achieves a higher effectiveness than the Random and Worst ofk approaches in maximizing the robustness of the image

				3	S	11/02/2022 14:41

The major findings in our research are as follows.
• DA techniques can improve the robustness of a model in image classification tasks by up to 0.73 pts for industrial GUI datasets. In particular, Sensei improves the robustness of the model by nearly 0.09 pts compared to the other techniques.
• There are many challenges in applying DA techniques to object detection tasks (e.g., some realistic variations for image classification incorrectly exclude the bounding boxes); they are highlighted in this study.
• Through feedback from developers, we identified two types of demand for a DA technique: (1) maintaining the training speed to avoid slowing the system development and (2) the extensibility for a variety of tasks (e.g., an anomaly detection

				4	S	11/02/2022 14:41

For the image classification task, theWorst ofk and Sensei methods achieved a better robustness for the image classification model than the Random approach. Therefore, we can answer 'yes' to RQ1 and confirm the effectiveness of the Worst ofk and Sensei approaches regarding the robustness of the image classification model with both open and industrial data. For the object detection task, the mAP and robustness of the
object detection model did not show any significant differences. Therefore, we must answer 'no' to RQ2 and cannot state that the Worst of k and Sensei methods contribute to the robustness in the object recognition task when applying the industrial GUI data used in this study. Through a manual investigation using our eyes, we found that some of the bounding boxes did not be learned properly by the object detection model when some or all parts of the bounding boxes extended outside the bounds of the image due to a translation or zoom in. In this case, the implicit conditions by which the human eye can correctly classify a sample were not satisfied. We believe we can solve this problem through one of the following two ways: (1) restricting the range of translation to prevent missing the bounding boxes

				5	S	11/02/2022 14:41

We asked the developers involved in the in-house ML system development for their feedback. Their opinions are as follows.
• A developer involved in creating an anomaly detection system from waveform data stated that such DA techniques can effectively generate anomaly data that are rarely observed in the real world.
• A developer who has already utilized the Random DA approach for improving the accuracy of a handwriting recognition system claimed that these DA techniques are expected to go beyond the Random method.
• A developer who uses an existing DA technique included in a deep learning framework is concerned that the improvement in robustness will result in a slower learning.
Through an in-house trial and interviews with in-house developers, we learned following lessons:
• The results of a long training time and feedback regarding the concerns of a comparatively slow system development indicate that DA techniques such as the Worst of k and Sensei approaches should not only to improve the robustness of the classifier but also to maintain the training speed.
• The findings regarding the difficulties in extending the DA techniques for an image classification task to an object detection task, and the feedback regarding the need for DA techniques in various domains indicate that the extensibility of the augmentation technique for a variety of tasks is important when designing a DA library

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On

Nodes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering\concept drift

PDF

Files\\An Empirical Study of the Impact of Data Splitting Decisions on the Performance of AIOps Solutions

No	Google Scholar	0.0115	5			
				1	S	10/02/2022 11:03

Handling concept drift. Existing AIOps studies usually train a static model regardless of po-
tential concept drift [5, 16, 24, 57, 71, 91] without respecting that the operation data is constantly evolving [17, 49, 51]. However, concept drift may lead to the obsolescence of such static models trained on previous data. To mitigate the impact of concept drift, other prior works suggest that AIOps models need to be retrained periodically to ensure that the models are not outdated

				2	S	10/02/2022 11:03

In this work, we first analyze the existence of concept drift in the studied operation dataset
(RQ3), then we evaluate the impact ofperiodically updating amodel instead ofusing a static model on the model performance, and how the model update frequency might impact the performance and cost of AIOps models (RQ4).

				3	S	10/02/2022 11:07

Concept drift exists in the operation data, which can be explained by the fact that the relationship between the variables in the operation data evolves over time. Practitioners and researchers should proactively detect and address the problem ofconcept drift in their AIOps

				4	S	24/02/2022 09:08

Periodically updated AIOps models provide better performance than the stationary
models, which suggest modelers update their AIOps models periodically. As shown in Figure 15(a) and Figure 15(b), periodically updating the models achieve a better performance in terms of the evaluated metrics than using a stationary model, and overall the difference becomes bigger when the distance between the training periods of the stationary model and the testing period becomes larger. We observe that the stationary models and the periodically updated models have a bigger difference on the Google dataset than on the Backblaze dataset, which may be explained by the fact that the Google dataset has a more severe concept drift issue (as discussed in

				5	S	24/02/2022 09:08

Increasing the frequency ofupdating the AIOps models can improve the performance,
while the improvement shows difference across models and datasets. Figure 16 shows the overall performance ofthe models (in terms ofAUC)whenwe vary the number oftime periods (i.e., N). Other performance metrics show a similar trend. In most cases, increasing the model update frequency can gradually improve model performance. For example, the AUC of the CART model on the Backblaze dataset increases by 3.5% (i.e., from 0.85 to 0.88) when we increase the number of time periods from 4 to 24 (the AUC of the stationary model, i.e., when N = 2, is 0.84). However, in some cases, for example, when updating the SVM model on the Backblaze dataset, we did not observe any performance improvement. We infer that some models (e.g., RGF) can not learn the evolving patterns in the datasets, thus are less sensitive to the update frequency. Increasing themodel update frequency increase the overall cost ofAIOps models; how-
ever, the cost increase varies significantly across models and datasets. Figure 17 shows the

Files\\Driftage~ a multi-agent system framework for concept drift detection

No	Web of science	0.0540	5			
				1	S	07/02/2022 22:38

This article proposes to create Driftage: a new framework using multi-agent systems to simplify the implementation of concept drift detectors considerably and divide concept drift detection responsibilities between agents, enhancing explainability of each part of drift detection. As a case study, we illustrate our strategy using a muscle activity monitor of electromyography. We show a reduction in the number of false-positive drifts detected, improving detection interpretability, and enabling concept drift detectors' interactivity with

				2	S	07/02/2022 22:38

There are many types of drifts in the concept drift detection
(CDD) area [7, 9, 10].

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On
				3	S	07/02/2022 22:39

supervised [13, 14], semi-supervised [15], unsupervised [16, 17, 18], statistical [19, 20], or even evolutionary algorithms [21] to deal with these drifts, but none of them is perfect for all drift types.
Some publications are arising with machine learning ensem-
bles for CDD because of the nature of the data that these detectors need to adapt to [22–25]. There are several factors such as data seasonality or change of data drift type; these ensembles can choose the best estimator for each case, and each estimator can still act alone. Nevertheless, this approach necessitates retraining of base learners and strategies to select

				4	S	07/02/2022 22:41
				6	S	07/02/2022 22:41

One of the most famous CDD algorithms is ADWIN (adaptive sliding window algorithm) [52]. It efficiently keeps a variablelength window of recent items, whose contents can be compared to discern whether there has been any change in the data distribution. This window is further divided into 2 subwindows (W0, W1) used to determine whether a change has happened. ADWIN compares the average ofW0 andW1 to confirm that they correspond to the same distribution. Concept drift is detected if the distribution equality no longer holds. Upon detecting a drift, W0 is replaced by W1 and a newW1 is initialized. ADWIN uses a

Nodes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering\HPO

PDF

Files\\Auptimizer -- an Extensible, Open-Source Framework for Hyperparameter Tuning

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On
No	Web of science	0.0154	1			
				1	S	08/02/2022 13:48

Auptimizer design goals are focused on a user-friendly in-
terface. Auptimizer benefits both practitioners and researchers and its design simplifies the integration and development of HPO algorithms. Specifically, the framework design helps both users to easily use Auptimizer in their workflows and researchers to quickly implement novel HPO algorithms. To reach these goals, the Auptimizer design has fulfilled the following requirements: • Flexibility. All implemented HPO algorithms share the same interface. This enables users to switch between different algorithms without changes in the code. A pool of HPO algorithms is integrated into the Auptimizer for users to explore and for researchers to benchmark against.
• Usability. Changes to existing user's code are limited to a minimal level. It reduces the friction for users to switch to the Auptimizer framework.
• Scalability. Auptimizer can deploy to a pool of computing resources to automatically scale out the experiment, and users only need to specify the resource.
• Extensibility. New HPO algorithms can be easily integrated into the Auptimizer framework if they followed the specified interface (see Section III-A).
Auptimizer addresses a critical missing piece in the appli-
cation aspect of HPO research. It provides a universal platform to develop new algorithms efficiently. More importantly, Auptimizer lowers the barriers for data scientists in adopting HPO into their practice. Its scalability helps users to train their models efficiently with all computing resources available. Switching between different HPO algorithms is simple and only needs changing the proposer name

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On

Files\\HyperNOMAD~ Hyperparameter Optimization of Deep Neural Networks Using Mesh Adaptive Direct Search

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On
No	ACM Digital library	0.0560	10			
				1	S	07/02/2022 16:19

The performance ofdeep neural networks is highly sensitive to the choice ofthe hyperparameters that define the structure of the network and the learning process. When facing a new application, tuning a deep neural network is a tedious and time-consuming process that is often described as a “dark art.” This explains the necessity of automating the calibration of these hyperparameters. Derivative-free optimization is a field that develops methods designed to optimize time-consuming functions without relying on derivatives. This work introduces the HyperNOMAD package, an extension of the NOMAD software that applies the MADS algorithm [7] to simultaneously tune the hyperparameters responsible for both the architecture and the learning process of a deep neural network (DNN).

				2	S	07/02/2022 16:21

The hyperparameters that define a deep neural network can be separated into two categories: the
ones that define the architecture ofthe network and the ones that affect the optimization process of the training phase. Tuning the hyperparameters ofthe first category alone has led to a separate field of research called Neural Architecture Search (NAS) [25] that allowed achievement of state-ofthe-art performance [53, 65] on some benchmark problems, although at a massive computational cost of800 GPUs for a fewweeks. Typically, one would perform an NAS first and then start tuning the other hyperparameters with the optimized architecture. However, Zela et al. [64] argue that this separation is not optimal since the two aspects are not entirely

				3	S	07/02/2022 16:21

One of the first scientific approaches used to tackle the HPO problem of neural networks is
grid search. This method consists of discretizing the hypercube defined by the range of each

				4	S	07/02/2022 16:20
				5	S	07/02/2022 16:21

Genetic algorithms are evolutionary heuristics that are also used for the HPO problem. Inspired
by biology, a genetic algorithm generates an initial population, i.e., a set ofconfigurations. Then, it combines the best parents to create a new generation of children. It also introduces random mutations to ensure a certain diversity in the population. These heuristics are therefore adaptive, thus exploring the space more wisely even ifsome randomness remains in the process. These algorithms are often used to optimize hyperparameters [26, 57, 63]. In [45], a method based on particle swarm optimization is able to provide networks with higher performance than those defined by experts in less time than what would have required a grid search or a completely random search. Another approach using the evolutionary algorithm CMA-ES [46] was proposed with satisfactory results.

				6	S	07/02/2022 16:21

Bayesian optimization (BO) can be seen as a subclass of DFO methods and, as such, can
be used to solve the HPO problem. The BO methods use information collected during previous assessments to diagnose the search space and predict which areas to explore first. Among them, Gaussian processes (GPs) are models that seek to explain the collected observations that supposedly come from a stochastic function. GPs are a generalization ofmulti-variate Gaussian distributions, defined by a mean and a covariance function. GPs are popular models for optimizing the hyperparameters ofneural networks [56, 60]. However, the disadvantage ofGPs is that they do not fit well to categorical features, and their performance depends on the choice of the kernel function that defines them. Tree-structured Parzen Estimator (TPE) is also a Bayesian method that can be used as a model instead of a GP.

				7	S	07/02/2022 16:21
				8	S	07/02/2022 16:22

The HyperNOMAD package is available onGitHub.1 It contains a series ofPython modules that act as a blackbox, which takes a set ofhyperparameters described in Section 3 as inputs and constructs the corresponding network that is trained and tested before returning the test accuracy as the output. This blackbox uses the PyTorch package [48] for its simplicity. HyperNOMAD also contains an interface that runs the optimization ofthe blackbox using the NOMAD software [37] described in the rest of this section. The basic usage of HyperNOMAD is described in Appendix A.

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On
				9	S	07/02/2022 16:23

4.1 Overview of NOMAD TheNOMADsoftware [37]isa C++ implementation oftheMADS algorithm [7, 9], which is a direct search method that generates, at each iteration k, a set ofpoints on the meshMk = {x + diag(δk)z : x ∈ Vk, z ∈ Zn },where Vk contains the points that were previously evaluated (including the current iterate xk)and δk ∈ Rn is the mesh size vector. Each iteration of MADS is divided into two steps: the search and the poll.The search phase
is optional and can contain different strategies to explore a wider space in order to generate a finite number of possible mesh candidates. This step can be based on surrogate functions, Latin hypercube sampling, and so forth [4, 11]. The poll, on the other hand, is strictly defined since the convergence theory ofMADS relies entirely on this phase. In the poll step, the algorithm generates directions around the current iterate xk to search for candidates locally in a region centered around xk and of radius, in each dimension, of Δk ∈ Rn, which is called the poll size vector.The setof candidates in this step defines the poll set Pk. IfMADS finds a better point than the incumbent, then the iteration is declared a success, and the
mesh and poll sizes are increased. However, if the iteration fails, then both parameters are reduced so thatδk ≤ Δk is maintained. This relation ensures that the set ofsearch directions becomes dense in the unit sphere asymptotically. In addition, NOMAD can handle categorical variables by adding a step in the basic MADS algorithm. A variable is categorical when it can take a finite number of nominal or numerical values that express a qualitative property that assign the variable to a class (or category). The algorithm relies on an ad hoc neighborhood structure, provided in practice by the user as a list ofneighbors for any given point. The poll step ofMADS is augmented with the socalled extended poll that links the current iterate xk with the independent search spaces where the neighbors can be found. The first neighbor that improves the objective function is chosen and the optimization carries on in the corresponding search space.

				10	S	07/02/2022 16:23

The selected neighborhood structure in HyperNOMAD relies on blocks of categorical variables with their associated variables. The following subsections describe this structure.
4.2.1 Blocks ofHyperparameters. HyperNOMAD splits the hyperparameters (HPs) defined
in Section 3.1 into different blocks: one for the convolution layers, the fully connected layers, and the optimizer and one for each of the other HPs. A block is an implemented structure that stores a list ofvalues, each one starting with a header and followed by the associated variables, when applicable, that are gathered into groups. For example, consider a CNN with two convolutional layers, each one defined with the number of output channels, the kernel size, the stride, the padding, and whether a pooling is applied or not as stated in Table 2. Then consider the values (16, 5, 1, 1, 0) and (7, 3, 1, 1, 1). Each set ofvalues corresponds to a group ofvariables that describes one convolutional layer and both groups are part of the convolution block. The header of the convolution block is the categorical variable that represents the number ofconvolutional layers (n1) that the CNN contains as showninFigure 5 (top).

Files\\Tunability~ Importance of Hyperparameters of Machine Learning Algorithms

No	Scopus	0.0579	6			
				1	S	11/02/2022 14:54

Modern supervised machine learning algorithms involve hyperparameters that have to be set before running them. Options for setting hyperparameters are default values from the software package, manual configuration by the user or configuring them for optimal predictive performance by a tuning procedure. The goal of this paper is two-fold. Firstly, we formalize the problem of tuning from a statistical point of view, define data-based defaults and suggest general measures quantifying the tunability of hyperparameters of algorithms. Secondly, we conduct a large-scale benchmarking study based on 38 datasets from the OpenML platform and six common machine learning algorithms. We apply our measures to assess the tunability of their parameters. Our results yield default values for hyperparameters and enable users to decide whether it is worth conducting a possibly time consuming tuning strategy, to focus on the most important hyperparameters and to choose adequate hyperparameter spaces for tuning.

				2	S	11/02/2022 14:55

The best hyperparameter value for one parameter i on dataset j, when all other parameters are set to defaults from θ⍰ := (θ⍰
1, ..., θ⍰ θ(j)⍰ k), is denoted by
i := arg min θ∈Θ,θl=θ⍰
R(j)(θ). l ∀ l⍰=i A natural measure for tunability of the i-th parameter on dataset j is then the difference
in risk between the above and our default reference configuration: d(j)
Furthermore, we define d(j),rel i
i := R(j)(θ⍰) −R(j)(θ(j)⍰ d(j)
= i), for j = 1, ..., m, i = 1, ..., k. (6)
d(j) as the fraction of performance gain, when we only i
tune parameter i compared to tuning the complete algorithm, on dataset j. Again, one can calculate the mean, the median or quantiles of these two differences over the n datasets, to get a notion of the overall tunability di of this parameter.

				3	S	11/02/2022 14:55

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On
				4	S	11/02/2022 14:56

Tunability of Hyperparameter Combinations and Joint Gains As an example, Table 4 displays the average tunability di1,i2
of all 2-way hyperparameter
combinations for rpart. Obviously, the increased flexibility in tuning a 2-way combination enables larger improvements when compared with the tunability of one of the respective individual parameters. In Table 5 the joint gain of tuning two hyperparameters gi1,i2
instead
of only the best as defined in Section 3.5 can be seen. The parameters minsplit and minbucket have the biggest joint effect, which is not very surprising, as they are closely related: minsplit is the minimum number of observations that must exist in a node in order for a split to be attempted and minbucket the minimum number of observations in any terminal leaf node. If a higher value of minsplit than the default performs better on a dataset it is possibly not enough to set it higher without also increasing minbucket, so the strong relationship is quite clear. Again, further figures for other algorithms are available through the shiny app. Another remarkable example is the combination of sample.fraction and min.node.size in ranger: the joint gain is very low and tuning sample.fraction only seems to be enough, which is concordant to the results of Scornet (2018). Moreover, in xgboost the joint gain of nrounds and eta is relatively low, which is not surprising, as these parameters are highly connected with each other (when setting nrounds higher, eta should be set lower and vice versa).
5.5. Hyperparameter Space for Tuning
The hyperparameter space for tuning, as defined in Equation (10) in Section 3.6 and based on the 0.05 and 0.95 quantiles, is displayed in

				5	S	11/02/2022 14:56

Our paper provides concise and intuitive definitions for optimal defaults of ML algorithms and the impact of tuning them either jointly, tuning individual parameters or combinations, all based on the general concept of surrogate empirical performance models. Tunability values as defined in our framework are easily and directly interpretable as how much performance can be gained by tuning this hyperparameter?. This allows direct comparability of the tunability values across different algorithms. In an extensive OpenML benchmark, we computed optimal defaults for elastic net, deci-
sion tree, k-nearest neighbors, SVM, random forest and xgboost and quantified their tunability and the tunability of their individual parameters. This—to the best of our knowledge— has never been provided before in such a principled manner. Our results are often in line with common knowledge from literature and our method itself now allows an analogous analysis for other or more complex methods. Our framework is based on the concept of default hyperparameter values, which can be
seen both as an advantage (default values are a valuable output of the approach) and as an inconvenience (the determination of the default values is an additional analysis step and needed as a reference point for most of our measures). We now compare our method with van Rijn and Hutter (2017). In contrast to us, they
apply the functional ANOVA framework from Hutter et al. (2014) on a surrogate random forest to assess the importance of
hyperparameters regarding empirical performance of a support vector machine, random forest and adaboost, which results in numerical

				6	S	11/02/2022 14:56

scores for individual hyperparameters. Their numerical scores are - in our opinion - less directly interpretable, but they do not rely on defaults as a reference point, which one might see as an advantage. They also propose a method for calculating hyperparameter priors, combine it with the tuning procedure hyperband, and assess the performance of this new tuning procedure. In contrast, we define and calculate ranges for all hyperparameters. Setting ranges for the tuning space can be seen as a special case of a prior distribution - the uniform distribution on the specified hyperparameter space. Regarding the experimental setup, we compute more hyperparameter runs (around 2.5 million vs. 250000), but consider only the 38 binary classification datasets of OpenML100 while van Rijn and Hutter (2017) use all the 100 datasets which also contain multiclass datasets. We evaluate the performance of different surrogate models by 10 times repeated 10-fold cross-validation to choose an appropriate model and to assure that it performs reasonably well.

Files\\Ultron-AutoML~ an open-source, distributed, scalable framework for efficient hyper-parameter optimization

No	IEEE	0.0266	2			
				1	S	03/02/2022 10:04

We present Ultron-AutoML, an open-source, dis-
tributed framework for efficient h yper-parameter optimization (HPO) of ML models. Considering that hyper-parameter optimization is compute intensive and time-consuming, the framework has been designed for reliability – the ability to successfully complete an HPO Job in a multi-tenant, failure prone environment, as well as efficiency – c ompleting t he j ob w ith minimum compute cost and wall-clock time. From a user’s perspective, the framework emphasizes ease of use and customizability. The user can declaratively specify and execute an HPO Job, while ancillary tasks – containerizing and running the user’s scripts, model checkpointing, monitoring progress, parallelization – are handled by the framework.

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On
				2	S	03/02/2022 10:09

Hyperparameter Optimization (HPO), also referred to as AutoML in the literature, can be cast as the optimization of an unknown, possibly stochastic, objective function mapping the hyper-parameter search space to a real valued scalar, the ML model's accuracy or any other performance metric on the validation dataset. The search-space can extend beyond algorithm or architecture specific elements to encompass the space of data pre-processing and data-augmentation techniques, feature selections, as well as choice of algorithms. This is sometimes referred to as the CASH (Combined Algorithm Search and Hyper-parameter tuning) problem for which algorithms have been proposed [28], [48]. Neural Architecture Search (NAS) is a special type of
HPO where the focus is on algorithm driven design of neural network architecture components or cells [26]. Models trained with architectures composed of these algorithmically designed neural network cells have been shown to outperform their hand-crafted counterparts in image recognition, object detection [57], and semantic segmentation [21], underscoring the practical importance of this field. Random Search [18] and Grid Search are effective HPO
strategies when the computational budget is limited or the hyper-parameter search space is high dimensional. Both are easy to implement and completely parallelizable. Random Search is also widely regarded as a good baseline for benchmarking new hyper-parameter optimization algorithms [33]. Bayesian Optimization (BO) is a dominant paradigm for
HPO [20], [27], [45]. Here, the objective function is modeled as a Gaussian Process [50], with the Kernel design reflecting assumptions about the objective function's smoothness properties. Under this assumption, the posterior distribution of the validation score for a candidate architecture is a Gaussian

Nodes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering\ML monitoring

PDF

Files\\Overton~ A Data System for Monitoring and Improving Machine-Learned Products

No	Google Scholar	0.0062	1			
				1	S	04/02/2022 22:03

In summary, Overton represents a first-of-its kind machine-learning lifecycle management system that has a focus
on monitoring and improving application quality. A key idea is to separate the model and data, which is enabled by a code-free approach to deep learning. Overton repurposes ideas from the database community and the machine learning community to help engineers in supporting the lifecycle of machine learning toolkits. This design is informed and refined from use in production systems for over a year in multiple machine-learned products.

Files\\Software Logs for Machine Learning in a DevOps Environment

No	Scopus	0.0653	5			
				1	S	11/02/2022 14:26

In this paper, we present the main challenges of contemporary approaches to generating, storing and managing the evolution of system logs data for large, complex, software-intensive systems based on an in-depth case study at a world-leading telecommunications company. Second, we present an approach for generating and managing the evolution of log data in a DevOps environment that does not suffer from the aforementioned challenges and is optimized for use in machine learning. Third, we provide validation of the approach based on expert interviews that confirm that the approach addresses the identified challenges and problems.

				2	S	11/02/2022 14:28

IV. SYSTEM LOGS FOR MACHINE LEARNING To address the challenges of using system logs for ML,
we have developed a novel approach consisting of three main parts. First, we discuss the DevOps scenario that logs optimized for ML could be applied to and the success factors which would emerge in it. Second, we propose the technical realization. Finally, we present the required process changes for realizing the proposed approach in an industrial context.
A. Logging in a DevOps Environment Based on our research at the case study company, as well as
experience from other companies, we identified three distinct

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On
				3	S	11/02/2022 14:28

DevOps scenarios, but not for autonomous system deployments. Finally, we need to govern evolution and backward compatibility in response to a constant flow of change requests from the R&D teams, customers, data scientists and others. The evolution of the system log entry model needs to be carefully managed as allowing for breaking changes may also invalidate data sets predating the change due to changes in semantics and/or structure. In the cases where introducing breaking changes is unavoidable, it may be necessary to develop mapping functions that allow for the generation of data sets that are based on system logs both before and after the breaking change. As virtually all machine learning algorithms perform better with a greater quantity of data, it is frequently beneficial to combine multiple logs into one data set for training and validation.

				4	S	11/02/2022 14:29

As presented in this section and in figure 4, generating logs
for machine learning requires that engineers, R&D teams and the organization change the way log entries are generated. However, the process by which system logs for machine learning are generated is, in principle, no more difficult than adding a normal log statement. The main difference is the organizational alignment and agreement on the structure and semantics of log entries. As usual, although most of the attention is quickly drawn towards the technical framework, it is the introduction of new processes and activities that will require the most effort and attention. Especially early in the process of adopting the approach outlined in this paper, it is beneficial to add a new AI log statement in the code at every place that there is an existing log statement, leaving the existing log statement. In this way, it is possible to generate two separate logs: the original log and the new AI log. The information that is presented in the AI log statement should be an encoded/normalised version of what is presented in the human readable log. In the case that a machine learning algorithm finds an anomaly in the AI log, the link with the human understandable entry in the human-readable log significantly helps the investigation into the detected anomaly.

				5	S	11/02/2022 14:29

Nodes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering\Model deployment

PDF

Files\\A comprehensive study on challenges in deploying deep learning based software

No	ACM Digital library	0.0144	5			
				1	S	11/02/2022 13:20

To fill this knowledge gap, this paper presents a comprehensive study on understanding challenges in deploying DL software. We mine and analyze 3,023 relevant posts from Stack Overflow, a popular Q&A website for developers, and show the increasing popularity and high difficulty of DL software deployment among developers. We build a taxonomy of specific challenges encountered by developers in the process ofDL software deployment through manual inspection of 769 sampled posts and report a series ofactionable implications for researchers, developers, and DL framework vendors.

				2	S	11/02/2022 13:25
				3	S	11/02/2022 13:27

7 IMPLICATIONS Based on the preceding derived findings, we next discuss our insights and some practical implications for developers, researchers, and DL framework vendors.
7.1 Researchers
As demonstrated in our study

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On
				4	S	11/02/2022 13:27

7.2 Developers
(1) Targeted learning of required skills. DL software deployment lies in the interaction between DL and SE. Therefore, DL software deployment requires developers with solid knowledge of both fields, making this task quite challenging. Our taxonomy can serve as a checklist for developers with varying backgrounds, motivating the developers to learn necessary knowledge before really

				5	S	11/02/2022 13:28

7.3 Framework Vendors
(1) Improving the usability ofdocumentation.As shown in our results, many developers even have difficulty in the entire procedure of deployment (i.e., how to deploy DL software). For instance, such questions account for 13.4% in mobile deployment. As described earlier, developers often complain about the poor documentation in these questions, revealing that the usability [71] of relevant documentation should be improved. Specifically, DL framework

Files\\An Empirical Study on Deployment Faults of Deep Learning Based Mobile Applications

No	ACM Digital library	0.0576	11			
				1	S	10/02/2022 10:48

Deep learning (DL) is moving its step into a growing
number of mobile software applications. These software applications, named as DL based mobile applications (abbreviated as mobile DL apps) integrate DL models trained using large-scale data with DL programs. A DL program encodes the structure of a desirable DL model and the process by which the model is trained using training data. Due to the increasing dependency of current mobile apps on DL, software engineering (SE) for mobile DL apps has become important. However, existing efforts in SE research community mainly focus on the development of DL models and extensively analyze faults in DL programs. In contrast, faults related to the deployment of DL models on mobile devices (named as deployment faults of mobile DL apps) have not been well studied. Since mobile DL apps have been used by billions of end users daily for various purposes including for safety-critical scenarios, characterizing their deployment faults is of enormous

				2	S	10/02/2022 10:48

To fill in the knowledge gap, this paper presents the first comprehensive study to date on the deployment faults of mobile DL apps. We identify 304 real deployment faults from Stack Overflow and GitHub, two commonly used data sources for studying software faults. Based on the identified faults, we construct a fine-granularity taxonomy consisting of 23 categories regarding to fault symptoms and distill common fix strategies for different fault symptoms. Furthermore, we suggest actionable implications and research avenues that can potentially facilitate the deployment of DL models on mobile devices.

				3	S	10/02/2022 10:51

To fill in the knowledge gap, this paper presents the first comprehensive study on analyzing symptoms and fix strategies of deployment faults of mobile DL apps. Given the surging popularity of mobile DL apps, this study is of enormous importance. It can help in understanding what are the common deployment faults of mobile DL apps and how these faults are resolved in practice, so as to provide a high-level categorization that can serve as a guide for developers to resolve common faults and for researchers to develop tools for detecting and fixing deployment faults of the increasing mobile DL apps.

				4	S	10/02/2022 10:53

IV. RQ1: SYMPTOMS Fig. 3 presents the hierarchical taxonomy of deployment
fault symptoms of mobile DL apps. The taxonomy is organized into three-level categories, including a root category (i.e., Deployment Faults), five inner categories linked to stages in deploying DL models (e.g., Model Conversion), and 23 specific leaf categories (e.g., Model parse failure). Finding 1: We construct a taxonomy of 23 fault symptom
categories related to deploying DL models on mobile devices, indicating the diversity of deployment faults. For each category, the number in the top right corner refers
to the number of faults in it. Due to space limit, we address only frequent and non-trivial symptoms (i.e., #faults ≥ 3). For Data Preparation and Model Update

				5	S	10/02/2022 10:53

Besides the faults with explicit errors thrown during the
model conversion stage, sometimes developers get unexpected models even after model conversion appears to be successfully done. For example, developers may find that the number, shape, or format of input/output tensors of the model changes. We classify these cases into the category Unexpected model (A.11), accounting for 4.1% faults in Model Conversion.
Finding 2: Most (i.e., 48.4%) of deployment faults occur
during the model conversion stage, covering a wide spectrum of symptoms (i.e., 12 categories). Among these categories, unsupported operation is the most common, accounting for 31.3% of faults in this stage

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On
				6	S	10/02/2022 10:53

After building projects, developers can run mobile apps to
make it predictable. However, in this phase, many developers encounter Framework loading failure (B.2) and Model loading failure (B.3), which refer to the failures in loading DL frameworks and models respectively and account for a total of 36.8% of faults in DL Integration. What is more, developers may configure projects to make it able to use the GPU backend on mobile devices. However, some developers complain that they encounter the GPU delegate failure (B.4) when running mobile DL apps. B.4 represents 21.1% of faults in DL Integration.
Finding 3: Faults appearing in the DL integration stage
account for 12.5% of the total deployment faults and cover five symptom categories. A large proportion (34.2%) of these faults are

				7	S	10/02/2022 10:53
				8	S	10/02/2022 10:53

In addition to the faults that affect the output results, there
are also 25.5% of faults that have impact on the memory usage and inference speed of mobile DL apps. We use Memory issue (D.4) and Speed issue (D.5) to refer to the two types of faults. Specifically, Memory issue (D.4) includes symptoms such as out of memory, memory leak, failures in memory allocation, and segment faults; Speed issue (D.5) is mainly manifested as long latency time of making inference.
Finding 4: 36.2% of faults occur when mobile DL apps
make inference based on input data, covering six symptom categories. In particular, 35.5% of the faults in this stage are captured since developers observe unexpected results.

				9	S	10/02/2022 10:54
				10	S	10/02/2022 10:54
				11	S	10/02/2022 10:55

Finding 7: The fix strategies for faults in inference are
diverse. They cover many stages of the deployment process, including fixing data processing, fixing the model conversion stage (e.g., fixing/using quantization), fixing the DL integration stage (e.g., fixing API usage during DL integration), etc.

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On

Nodes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering\Model development

PDF

Files\\Bighead~ A Framework-Agnostic, End-to-End Machine Learning Platform

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On
No	IEEE	0.0315	2			
				1	S	08/02/2022 12:56

Many machine learning platforms have been developed at
various companies. We briefly overview some major works in this section. TFX [3] is an end-to-end machine learning platform developed by Google, which spans from prototyping to production. It exclusively supports TensorFlow [7] as the model framework. Kubeflow [8] is also developed at Google, focusing on serving models in Kubernetes. MLflow [4] is developed and open sourced by Databricks. It is integrated with several cloud service providers, such as AWS and Azure. H2O [5] is a open source machine learning platform implemented in JVM with API libraries in several languages. Skymind Intelligence Layer [9], built on top of DeepLearning4J, offers model serving and scalability in its enterprise edition. Several in-house platforms cover many aspects of the machine learning workflow, such as Uber's Michelangelo [6], Facebook's FBLearner Flow [10], and Groupon's Flux [11]. However, these platforms are internal and not yet open sourced. Data Robot [12] is a popular proprietary system that offers features for automated machine learning. Several systems like Polyaxon [13], Comet [14], and Atalaya [15] provides model serving. Cloud service providers offer systems that enable the building, serving, and management of models, including Amazon's SageMaker [16], Microsoft Azure Machine Learning

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On
				2	S	08/02/2022 12:57

Files\\Challenges in Deploying Machine Learning~ a Survey of Case Studies

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On
No	Google Scholar	0.0028	1			
				1	S	07/02/2022 23:37

In this study, we undertake a survey of these reports to capture the current challenges in deploying machine learning in production1. First, we provide an overview of the machine learning deployment workflow. Second, we review use case studies to extract problems and concerns practitioners have at each particular deployment stage. Third, we discuss cross-cutting aspects that affect every stage of the deployment workflow: ethical considerations, end users' trust and security.

Nodes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering\Model ML traceability

PDF

Files\\MSR4ML~ Reconstructing Artifact Traceability in Machine Learning Repositories

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On
No	Web of science	0.0758	9			
				1	S	07/02/2022 11:05

In this work, we propose a framework for automatic identification and traceability of links between data, code and ML model through Mining Software Repositories (MSR) techniques. Our tool combines static code analysis and mining commit data to identify ML, code and data artifacts, reconstruct links between them and retrieve commits that affect each end of the link. The objective is to increase productivity and the developers' awareness of their project through the recovered traceability.

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On
				2	S	07/02/2022 11:06

The primary objective of this work is to initiate a discussion
about the peculiarities of ML applications as software projects and emphasize the need to increase the awareness of the diverse development teams about their artifacts. In this paper, we propose to leverage static code analysis and mining software repository (MSR) techniques to recover links between code, data, and ML models and improve traceability in Git-based

				3	S	07/02/2022 11:06

MSR4ML (MSR for ML) is a framework for automatic identification and tracing of artifact usage in Git-based ML projects. It explores the code and the repository of a ML project to extract relevant information about artifact usage and retrieve links between them. It aims to provide a tool for reconstructing traceability in existing Git-based ML projects.

				4	S	07/02/2022 11:07

Traceability: Our traceability process takes advantage of Git
features and model-artifact links to monitor the evolution of a ML project. It can be used for many purposes such as model metadata extraction, model analysis, continuous integration, and others. Consider the following example of its application for model analysis. A developer may ask: ”What changes caused the model to perform worse?” The corresponding query can be adapted as: ”Retrieve all commits affecting the model and its artifacts between the current version of the model and the previous one.” Our framework will follow these steps to get the information: 1) Using the model filename, extract the exact time t of the previous commit modifying the model; 2) Retrieve all the artifacts that are linked with the model; 3) Extract all the commits modifying these artifacts from t until now; 4) Return the commits, classifying them according to the priority of the link between the artifact and the model.

				5	S	07/02/2022 11:07

Source code parser The source code parser is responsible for parsing code files
and obtain their Abstract Syntax Tree (AST) representation for further analysis. The output of the parser is an extended AST node representing the contents of a code file. The AST allows us to traverse the code’s labelled nodes and find specific elements, including method invocations, variable declarations and accesses, string literals and others. The resulting AST must be complete, so that if reversed engineered, it would produce the exact original code.

				6	S	07/02/2022 11:07

Artifact usage identifier The artifact identifier traverses the AST produced by the
code parser and identifies all the methods or functions that interact with files. The assumption is that methods interacting with files may reveal links between code and data files.

				7	S	07/02/2022 11:07

Artifact classifier This module is responsible for classifying artifacts into
different categories. We can distinguish between four main artifacts of ML project: data, configuration, code and models [1], [5].

				8	S	07/02/2022 11:08

Commit tracker The commit tracker (Figure 5) is responsible for tracking
the commits associated with each artifact in the project, by querying Git for the relevant commit data. While the basic logic simply allows the interaction with a Git repository, this can be extended with plug-ins to interact with other platforms, like GitHub or Bitbucket, or include other third party Git libraries.

				9	S	07/02/2022 11:07

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On

Nodes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering\Model publishing and serving

PDF

Files\\DLHub~ Simplifying publication, discovery, and use of machine learning models in science

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On
No	Google Scholar	0.0259	5			
				1	S	07/02/2022 22:51

There is a growing need for “learning systems” to support various phases in the ML lifecycle. While others have focused on supporting model development, training, and inference, few have focused on the unique challenges inherent in science, such as the need to publish and share models and to serve them on a range of available computing resources. In this paper, we present the Data and Learning Hub for science (DLHub), a learning system designed to support these use cases. Specifically, DLHub enables publication of models, with descriptive metadata, persistent identifiers, and flexible access control. It packages arbitrary models into portable servable containers, and enables low-latency, distributed serving of these models on heterogeneous compute resources. We show that DLHub supports low-latency model inference comparable to other model serving systems including TensorFlow Serving, SageMaker, and Clipper, and improved performance, by up to 95%, with batching and memoization enabled.

Reference Number	Coded By Initials	Modified On
2	S	07/02/2022 22:52

In this paper, we present the Data and Learning Hub for science (DLHub)
and outline initial experiences applying this learning system to science. While many learning systems focus on building and training ML models <14; 15; 3>, DLHub is a unique learning system that is designed to support the publication and serving of ML models in science. DLHub is implemented as a cloud-hosted service that allows researchers to deposit and share models of various types, in-

Reference Number	Coded By Initials	Modified On
3	S	07/02/2022 22:53

DLHub offers a unique model serving infrastructure that is capable of serv-
ing many different types of models on a range of distributed computing resources including clouds, clusters, and supercomputers. The serving infrastructure builds upon funcX <19>—a distributed Function-as-a-Service platform developed specifically to support remote and distributed execution of functions. DLHub implements a flexible pipeline that converts deposited models into servables—executable containers that implement a standard DLHub execution interface, irrespective of the model type, and includes the trained model, model components (e.g., training weights, hyperparameters), and dependencies (e.g., system or Python packages).

Reference Number	Coded By Initials	Modified On
4	S	07/02/2022 22:55
5	S	07/02/2022 22:57

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On

Nodes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering\Model training

PDF

Files\\500+ times faster than deep learning~ a case study exploring faster methods for text mining stackoverflow

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On
No	ACM Digital library	0.0610	7			
				1	S	11/02/2022 13:33

For example, recent results show that for finding related Stack Overflow posts, a tuned SVM performs similarly to a deep learner, but is significantly faster to train. This paper extends that recent result by clustering the dataset,
then tuning every learners within each cluster. This approach is over 500 times faster than deep learning (and over 900 times faster ifwe use all the cores on a standard laptop computer). Significantly, this faster approach generates classifiers nearly as good (within 2% F1 Score) as the much slower deep learning method. Hence we recommend this faster methods since it is much easier to reproduce and utilizes far fewer CPU resources. More generally, we recommend that before researchers release
research results, that they compare their supposedly sophisticated methods against simpler alternatives (e.g applying simpler learners to build local models).

				2	S	11/02/2022 13:33

This paper further extends the Fu et al. Using very simple widely
used data mining method (K-Means), we can train even faster that Fu et al. and 500 times faster than deep learning (and over 900 times faster if we use all the cores on a standard laptop computer). The core to our approach is (1) building multiple local models then (2) tuning per local model. This paper evaluates this divide and conquer approach by: (1) Exploring the Xu et al. task using SVMandK-nearest-neighbor (KNN) classifiers;
(2) Repeating step 1 using hyperparameter tuning– specifically, differential Evolution (DE)– to select control parameters for those learners;
(3) Repeats steps 1 and 2 using local modeling: i.e. clustering the data then apply tuning and learning to each cluster;

				3	S	11/02/2022 13:33

• RQ1: Can we reproduce Fu et al.'s results for tuning SVM with differential evolution (DE)? Our DE with SVM perform no worse than Fu et al.
• RQ2: How do the local models compare with global models in both tuned and untuned versions in terms model training time?
Local models perform comparably to their global model counterparts, but are 570 times faster in model training time.(To be precise, that 570 figure comes fromrunning on a single core. Ifwe distribute the execution cross the eight cores of a standard laptop computer, our training times become 965 times faster.)
• RQ3: How does the performance of local models compare with global models and state-of-the-art deep learner when used with SVM and KNN?
Local models performance very nearly as well

				4	S	11/02/2022 13:34

Based on these experiments and discoveries, our contribution and outcome from the paper are:
• A dramatically faster solution to the Stack Overflow text mining task first presented by Xu et al. This new method runs three orders of magnitude faster than prior work.
• Support for “not everything needs deep learning”; i.e. sometimes, applying deep learning to a problem may not be the best approach.
• Support for a simplicity-first approach; i.e. simple method like K-Means_DE_SVM can performs as good some of the state of the art models but with a (much) faster training time.
• Support for local modeling. Such local models can significantly reduce training time by clustering data then restricting learning to on each cluster.
• A reproduction package - which can be used to reproduce, improve or refute our results1.

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On
				5	S	11/02/2022 13:34

RQ1: Can we reproduce Fu et al.'s results for tuning SVM with differential evolution (DE)? This study uses same differential evolution with SVM for both
global and local models. Thus to compare with Fu's DE with SVM as global model, the first task as part ofthis experiment was to recreate Fu et al.'s work so that this study have a baseline to measure against. Hence, this research question is a "sanity check" that must be passed before moving on to the other, more interesting research questions. The study uses the same SVM from Scikit-learn with the pa- rameters tuned as mentioned in Table 2. Here the training time of the DE+SVM model is also compared with Fu et al.'s model. Table 6 shows the class by class comparison for all the performance measure this study is using. From Table 6 it can be seen that our results with SVM with DE
for hyperparameter tuning [9] [12] similar to the results of Fu et al. It can be observed from this figure that for most of the cases apart from class 3, the model has performed a little better, but the delta between the performance is very small. Hence the answer to our RQ1, is that this study has success-

				6	S	11/02/2022 13:34

RQ2: How do the local models compare with global models in both
tuned and untuned versions in terms model training time? For RQ2, this experiment built one model for each clusters using
either normal or tuned versions ofSVM or KNN (where tuning was performed with DE): For the default SVM and KNN the experiment uses the default parameters, described in Table 1. As discussed above, this study have used the GAP statistic [33] [45]
for finding the best number of clusters, using minimum and maximum number of clusters as 3 and 15, respectively. As part of the experiment we learned that 13 clusters achieves best results (measured as per the GAP statistic). This study measures the time taken for this model to train which
includes time taken by GAP statistic, K-Means training time, and SVM/KNN with DE training time. Figure 6 compare the model training time in log scale of all
models with the results from XU et al.'s CNN approach. Its apparent from the figure 6 that for this domain KNN and SVM has the fastest runtimes. That said, as describe below, we cannot recommend these methods since, as shown below, they achieve poor F1 Scores.

				7	S	11/02/2022 13:34

RQ3: How does the performance oflocal models compare with
global models and state-of-the-art Deep Learner when used with SVM and KNN? The final part of our research question was to check if the local
models performance is comparable to Fu et al.'s DE_SVM and the XU's state of the art CNN. To evaluate the performance of the models this study compares F1 performance measures described in Section 3.3. As mentioned in the section, a 10 fold * 10 repeat cross validation was performed, so all the results are mean of 100 models created.
Figure 7 shows our F1 Score results (mean result across all 4 class
of Table 6). The numbers on top of each bar show the results of statistical tests. Bars with the same rank are statistically indistinguishable. Note that these results should be discussed with respect to the runtime results shown above:

Files\\All versus one~ an empirical comparison on retrained and incremental machine learning for modeling performance of adaptable software

No	Scopus	0.0643	14			
				1	S	10/02/2022 11:26

This paper is the first to report on a comprehensive empirical study that examines both modeling methods under distinct domains of adaptable software, 5 performance indicators, 8 learning algorithms and settings, covering a total of 1,360 different conditions. Our findings challenge the general belief, which is shown to be only partially correct, and reveal some of the important, statistically significant factors that are often overlooked in existing work, providing evidence-based insights on the choice.

				2	S	10/02/2022 11:40

Incremental modeling is chosen for faster training [12] [13] [14] [8] while the retrained modeling is chosen when higher accuracy is preferred [15] [16] [17] [18] [9] [19] [20] [7] [14]. The choice is a tradeoff between accuracy and training time

				3	S	10/02/2022 11:40

RQ1: Does the retrained version of a given learning algorithm always make more accurate model than its incremental counterparts when modeling adaptable software? No it does not, the incremental modeling can achieve statistically better accuracy under certain learning algorithms, the adaptable software and the fluctuations of the obtained data, which is clearly contradict to what the general belief claims.

				4	S	10/02/2022 11:40

RQ2: Does the incremental version of a given learning algorithm constantly leads to faster training than its retrained counterparts when modeling adaptable software? Yes it does, as the general belief stated. However, the gain on training time may be practically trivial.

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On
				5	S	10/02/2022 11:40

RQ3: When choosing modeling methods considering different learning algorithms, do the trade-offs between accuracy and training time for modeling performance of adaptable software always needed?
Trade-off is indeed required, in which the incremental modeling could train faster but with worse accuracy. However, this is not always the case—it is possible that the incremental modeling achieves the best for both properties. Therefore, the general belief is inaccurate.

				6	S	10/02/2022 11:41

RQ4: How the modeling methods can be affected by the runtime fluctuations of the adaptable software, i.e., the number of concept drifts and the deviations in the data?
The errors of both modeling methods exhibit considerably positive monotonic correlations to the number of drifts, and non-trivial negative monotonic correlations to the deviations of data. We did not observe clear correlations of their training time to the number of concept drift and data deviations in general. The only exception is the strong correlation between training time of incremental modeling and the number of concept drift.

				7	S	10/02/2022 11:43
				8	S	10/02/2022 11:43
				9	S	10/02/2022 11:44
				10	S	10/02/2022 11:45

For RQ1, we obtained the following findings:
Finding 1: The retrained version of a given learning algorithm does not always lead to higher accuracy than its incremental counterpart. In fact, the winner on accuracy can be considerably affected by the actual learning algorithm, i.e., incremental modeling is better with MLP while the retrained one is better with SVM, and the characteristics of subject adaptable software, i.e., the incremental modeling is more accurate for highly fluctuated adaptable software while the retrained one is better for stable software. Finding 2: Overall, the retrained modeling tends to be more robust accuracy than that of the incremental modeling. This would affect the choice for adaptable software where the stability is more important than having greater accuracy. Finding 3: For ensemble learning algorithms, the incremental modeling has consistently better accuracy on Bagging while the retrained one shows less error on Boosting.

				11	S	10/02/2022 11:45

For RQ2, we have the following findings:
Finding 4: Although the incremental modeling has statistically shorter training time than that of the retrained one (from 15% to three order of magnitude), the practical improvement may be trivial depending on the learning algorithms, e.g., for MLP, this can be practically important but may be negligible for other learning algorithms. Finding 5: Training time of incremental modeling is more robust while that of the retrained one varies depending on the subject adaptable software: more stable software system can lead to robust training time while fluctuated ones can impose varied training time. This would affect the choice for adaptable software where any single spike of high training time can cause serious consequence.

				12	S	10/02/2022 11:45

For RQ3, we obtained the following findings: Finding 6: With all the learning algorithms studied, the incremental modeling yields better accuracy and training time for 3 out of the 5 performance indicators considered. For the remaining two indicators, there is a trade-off when considering all the learning algorithms studied: the incremental modeling could exhibit shorter training time but worse accuracy. Conversely, the retrained modeling tends to impose longer training time but lead to better accuracy. This means that it is possible for the incremental modeling to achieve the best on both accuracy and training time. Finding 7: Even for the same learning algorithm, the decision of using incremental or retrained modeling can be a trade-off, see for example the DT on throughput.

				13	S	10/02/2022 11:45

For RQ4, we obtained the following findings: Finding 8: For both the incremental and retrained modeling, their errors exhibit considerably positive monotonic correlations to the number of drifts, and non-trivial negative monotonic correlations to the deviations (mRSD) of data. Relatively, the accuracy of incremental modeling worse off faster when the number of drifts increase; and improve quicker when the mRSD becomes larger. Finding 9: For the incremental modeling, its training time has strong negative monotonic correlations to the number of drifts while the correlation between the training time of retrained modeling and the number of drifts is arbitrary. There is also no clear relationship between the training time of both modeling methods and the deviations (mRSD) of data, or such a relationship is

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On
				14	S	10/02/2022 11:45

Lesson 1: The original belief has flaws and is inaccu-
rate. Findings 1 - 3 are clear contradictions to the general belief when a learning algorithm is considered, such that the retrained modeling do not always lead to better accuracy than its incremental counterpart. Our findings have revealed some patterns when choosing the method, for example, the incremental modeling is more accurate for highly fluctuated adaptable software while the retrained one is better for stable software. The retrained modeling also exhibits more robust accuracy overall. Despite that the incremental modeling is always trained faster with better robustness than its retrained counterpart (Finding 4 and 5), which is consistent with the belief, the distinction may be practically insignificant, e.g., they differ only in milliseconds. Lesson 2: Trade-off between accuracy and training time exists, but not always. When considering all learning algorithms, tread-off is needed based on preferences, but not always. The findings (Finding 6 and 7) reveal that it is possible for the incremental modeling to perform better on both accuracy and training time; This is partially comply with the general belief. Lesson 3: Runtime fluctuation (i.e., number of drifts
and deviations of data) could indeed impose non-trivial monotonic impacts on the accuracy, but limited on training time of both modeling methods. Our empirical findings (Finding 8 and 9) reveal that, in contrast to the retrained modeling, the accuracy of incremental modeling exhibits generally stronger, monotonic correlations to the number of drifts

Nodes\\Maintainable ML\\Available solutions for maintaining a ML systems\ML Model Engineering\Testing

PDF

Files\\Automatic Unit Test Generation for Machine Learning Libraries~ How Far Are We~

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On
No	ACM Digital library	0.0280	8			
				1	S	08/02/2022 13:30

In this paper, we set out to investigate the effectiveness of exist-
ing unit test generation techniques on machine learning libraries. To investigate this issue, we conducted an empirical study on five widely -used machine learning libraries with two popular unit test case generation tools, i.e., EVOSUITE and Randoop. We find that (1) most of the machine learning libraries do not maintain a high-quality unit test suite regarding commonly applied quality metrics such as code coverage (on average is 34.1%) and mutation score (on average is 21.3%), (2) unit test case generation tools, i.e., EVOSUITE and Randoop, lead to clear improvements in code coverage and mutation score, however, the improvement is limited, and (3) there exist common patterns in the uncovered code across the five machine learning libraries that can be used to improve unit test case generation tasks.

				2	S	08/02/2022 13:32

In this paper, we set out to investigate the effectiveness
of the widely-used automatic unit test generation techniques on ML libraries. Specifically, we select five widely-used ML libraries, i.e., Weka [13], Stanford CoreNLP [14], Mallet [15], OpenNLP [16], and Mahout [17]. Additionally, to better understand ML libraries, inspired by existing studies [10, 12], we decompose a ML library into three different types of components, i.e., data process, core model, and util (Details are in Section II-B). We use two typical automatic unit test generation tools, i.e., EVOSUITE and Randoop, as the experiment objectives following prior studies [5, 6]. For our study, we first perform an empirical study on the
five ML libraries to unveil the effectiveness of their current unit test suites regarding commonly applied quality metrics such as code coverage and mutation score [18]. We then apply EVOSUITE and Randoop on these ML libraries to generate unit tests and check whether EVOSUITE and Randoop could improve test effectiveness on these libraries, regarding code coverage and mutation score, by comparing the automatically generated tests against the existing manually created ones.

				3	S	08/02/2022 13:32

This paper makes the following contributions: • We conduct a comprehensive investigation of current unit test practices on five widely-used machine learning libraries.
• We examine the effectiveness and usefulness of two widely-used automatic unit test generation tools on five machine learning libraries.
• We identify gaps between existing automatic unit test generation techniques and unit testing practices on machine learning libraries.
• We discuss general lessons learned and future directions from the application of the automatic unit test generation to machine learning libraries.
The rest of this paper is organized

				4	S	08/02/2022 13:34

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On
				5	S	08/02/2022 13:34

Current unit test suite in ML libraries has lower quality regarding code coverage (on average, 34.1%) and mutation score (on average, 21.3%). In addition, the testing effort of academic-led ML libraries is unbalanced distributed and their unit test quality is significantly worse than that of community-led ML libraries.

				6	S	08/02/2022 13:34

EVOSUITE and Randoop lead to clear improvements in code coverage and mutation score compared to the original unit test suites of ML libraries. However, on average, 45.4% code is still uncovered with the generated test cases.

				7	S	08/02/2022 13:34

Overall, the unit test suites in ML libraries mainly focus on a subset of valid functionalities. In addition, there exists common patterns among the uncovered code of the studied ML libraries.

				8	S	08/02/2022 13:34

Both EVOSUITE and Randoop can significantly help cover AUX and MEB, while the performance on other three categories, i.e., VB, IVB, and EX, is limited.

Files\\Automatically Authoring Regression Tests for Machine-Learning Based Systems

No	Web of science	0.0388	5			
				1	S	08/02/2022 13:22

We identify four specific challenges and address them by developing a new general methodology to automatically author and maintain tests. In particular, we use the volume of production data to periodically refresh our large corpus of test inputs and expected outputs; we use perturbation of the data to obtain coverage-adequate tests; and we use clustering to help identify patterns of failures that are indicative of software bugs. We demonstrate our methodology on an ML-based context-aware Speller. Our coverage-adequate, approx. 1 million regression test cases, automatically authored and maintained for Speller (1) are virtually maintenance free, (2) detect a higher number of Speller failures than previous manually-curated tests, (3) have better coverage of previously unknown functional boundaries of the ML component, and (4) lend themselves to automatic failure triaging by clustering and prioritizing subcategories of tests with over-represented failures. We identify several systematic failure patterns which were due to previously undetected bugs in the Speller, e.g., (1) when the user misses the first letter in a short word, and (2) when the user mistakenly inserts a character in the last token of an address;

				2	S	08/02/2022 13:25

In this paper, we develop a new methodology aimed at functional regression testing for ML software, and apply it to a context-aware ML-based spelling checker/corrector (Speller). Our results show that the methodology can scale up the test suite to cover a large typo space, and, at the same time, reveal failure cases that can often be masked by other common misspelled inputs. Furthermore, the methodology can cover a multidimensional space with test cases automatically built upon constantly-changing production data. We show that these adaptive test suites can isolate the performance of the ML component from the end-to-end speller system, and keep up with both model

				3	S	08/02/2022 13:26

Instead of reporting individual test failures, we rely on featurizing misspell patterns and spectral clustering to automatically report subcategories of tests that contain higher proportions of defects. In particular, we make the following contributions: • We keep up with the ML software's evolving input space by automatically revising our test suite using production data to obtain new test cases and delete obsolete ones. • We learn a coverage-driven perturbation model to generalize existing cases in production data to enrich edge cases that are underrepresented in real training and test data. • We resolve the obsolete oracle problem by using the relationship between the original data from production and its perturbed counterparts; we determine the expected output of a number of test cases where the consequent feedback is positive and indicates that the users received correct outputs. • We automatically identify important failure classes by mining patterns of test cases using unsupervised learning and cluster the test cases to identify subgroups with high number of

				4	S	08/02/2022 13:26

METHODOLOGY TO TEST ML-BASED SYSTEMS We have developed a new methodology to address the chal-
lenges brought about by ML systems. The key intuition behind our methodology is to leverage the scale of production data to automatically author large numbers of coverage-adequate test cases whose Pass/Fail outcomes reveal systematic patterns that may be indicative of failures in the ML system. More specifically, as shown in Figure 1, we start by mining (A) large volumes of production data, which we then perturb using (B) a coverage-driven model-based test input curator that yields a large number of coverage-adequate test cases, with test inputs and expected outputs. These tests are then executed on the ML SUT, the actual output is obtained, and (C) an automated test oracle determines if the SUT passed or failed the test. Together with features of the inputs, test outcomes, and expected outputs, we use (D) clustering to determine which failures are related, in that all failures in a given cluster stem from a single bug. The results from clustering can also be used to improve the curator to generate more refined test suites along with oracles. We now break

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On
				5	S	08/02/2022 13:26

Files\\Cats are not fish~ deep learning testing calls for out-of-distribution awareness

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On
No	ACM Digital library	0.0465	7			
				1	S	08/02/2022 12:36

Although recent progress has been made in designing novel testing techniques for DL software, which can detect thousands of errors, the current state-of-the-art DL testing techniques usually do not take the distribution of generated test data into consideration. It is therefore hard to judge whether the "identified errors" are indeed meaningful errors to the DL application (i.e., due to quality issues of the model) or outliers that cannot be handled by the current model (i.e., due to the lack of training data). Tofill this gap, we take thefi rst step and conduct a large scale empirical study, with a total of 451 experiment configurations, 42 deep neural networks (DNNs) and 1.2 million test data instances, to investigate and characterize the impact of OOD-awareness on DL testing. We further analyze the consequences when DL systems go into production by evaluating the effectiveness of adversarial retraining with distribution-aware errors. The results confirm that introducing data distribution awareness in both testing and enhancement phases outperforms distribution unaware retraining by up

Reference Number	Coded By Initials	Modified On
2	S	08/02/2022 12:41

We select 5 state-of-the-art OOD-detection techniques that are commonly used among related literature [4, 16, 23, 24, 36, 37, 40]. OOD techniques use different approaches to retrieve an OOD score. Some use input perturbation, and others require a specifically trained new DNN. Therefore, this work includes techniques with various approaches as follows: • Simple Baseline [15]. The baseline identifies that in and outof-distribution samples are classified with different probability distributions. The softmax prediction probability is used to determine whether an input is ID or OOD.

• ODIN [24]. In addition to calculate the softmax prediction probability proposed by the baseline, ODIN adds temperature scaling to the input as well as small input perturbations. They show that small perturbations have stronger effects on in-distribution samples rather than out-of-distribution samples, achieving higher ID/OOD classification performance.

• Mahalanobis [23]. Mahalanobis detection technique integrates the information from all layers into the score calculation. It takes the closest class for each layer, adds small noise to the test sample andfi nally computes the score by measuring the Mahalanobis distance [29] between the test sample and the closest class-conditional Gaussian distribution.

• Outlier Exposure [16]. Outlier Exposure stands out by classifying inputs with a separately trained DNN which is exposed to the same training data as the DNN used for the application. However, in addition, out-of-distribution data is integrated into the training procedure of the outlier exposure DNN model. Afterward, the maximum softmax probability is taken similar to the baseline for out-of-distribution detection.

• Likelihood-Ratio [40]. The latest contribution of thefi eld utilizes a separately trained DNN, namely a generative DNN model with

Reference Number	Coded By Initials	Modified On
3	S	08/02/2022 12:41

Overall, our results show that Outlier Exposure on Densenet-121 architecture performs the best and the results are consistent on all benchmark datasets. The existing techniques can detect the ID data effectively where most of the test data are correctly classified as in-distribution. Splitting the classes of the training set imposes a challenge to the detection techniques and grants a new perspective on their performance for application-realistic settings.

Reference Number	Coded By Initials	Modified On
4	S	08/02/2022 12:42

Answer to RQ2: The data distribution generated by mutation operators is dependent on the datasets. Considering the same mutation operators, more test cases tend to be more OOD for grayscale images and less for color images. Image blur and Image Scale are the mutations strategies where the highest OOD-score is observed, whereas Image Rotation, Shear, Brightness and Contrast generate fewer OOD data. The error test cases are more likely to be OOD than benign test cases.

Reference Number	Coded By Initials	Modified On
5	S	08/02/2022 12:42

Answer to RQ3: Our results show that, existing coverage criteria affect the data distribution of generated test cases, which is important to address when designing a test scenario. KMNC, TKNC, NC and FANN tend to decrease the number of OOD benign test cases while NC and NBC tend to increase the OOD benign test cases. For the mutation operators that tend to generate fewer OOD data such as rotation and contrast, the existing coverage criteria can increase the number of OOD data by covering more behaviors of the DNN. For the mutation that tends to generate more OOD data such as blur, the existing coverage criteria can decrease the number byfiltering some data with the coverage guidance. For grayscale images, the coverage criteria may decrease the number ofOOD data with random mutation operators. The coverage criteria may increase the OOD data for generated error test cases.

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On
				6	S	08/02/2022 12:42

Answer to RQ4: The results demonstrate that ID-errors tend to befi xed via DNN adjustments, while OOD-errors seem to require further training data for being correctly classified. When retraining, OOD errors tend to be on average 10.4% more effective in improving the robustness of the DNN than ID errors or randomly chosen ones. Furthermore, not all OOD errors help the model to generalize, indicating that the OOD-score distance towards the trained/tested DNN distribution matters when choosing the right data for enhancing robustness.

				7	S	08/02/2022 12:42

• OOD Detection for DL Testing (RQ1). In DL testing, it is still challenging to distinguish ID and OOD data especially when more similarities between the two tested data types exist. Therefore,fi ne-grained thresholds seem helpful in gaining a better understanding in similar cases. Our results in Fig. 2 provide the following guidance: if the testing tool aims at generating ID test cases, a smaller N should be selected. If we want to generate OOD test cases, a larger N should be selected. Research Guidance: a possible direction is to develop OOD techniques, which can effectively detectfi ne-grained OOD data for deep learning testing.
• Mutation Operators and Coverage Criteria (RQ2&3). Our results show that the existing mutation and coverage criteria have different effects on ID data or OOD data generation. To build the distribution-aware DL testing tools, we could develop distribution-based coverage criteria that canfi lter some OOD data or ID data. Research Guidance: DL testing tools should be aware of distribution. A promising direction is to develop morefi ne-grained distribution-aware criteria for the test selection.
• Robustness Enhancement (RQ4.) Our initial results have shown that distribution-aware retraining is more effective in robustness enhancement than the distribution-unaware retraining. It seems that root causes for ID errors are partially model dependent while OOD errors can be effectivelyfi xed with new training data. Research Guidance: A future research direction is to further analyze the root cause of ID and OOD errors, especially in an even morefi ne grained setting which can provide guidance for repairing the model from a data and DNN architecture perspective under regard of the presented threshold of this work.

Files\\Machine Learning Testing~ Survey, Landscapes and Horizons

No	Google Scholar	0.0131	5			
				1	S	07/02/2022 12:59

For example, DeepXplore [1], a differential white-box
T
testing technique for deep learning, revealed thousands of incorrect corner case behaviours in autonomous driving learning systems; Themis [5], a fairness testing technique for detecting causal discrimination, detected significant ML model discrimination towards gender, marital status, or race for as many as 77.2 percent of the individuals in datasets to which it was applied.

				2	S	07/02/2022 13:01
				3	S	07/02/2022 13:02
				4	S	07/02/2022 13:04
				5	S	07/02/2022 13:07

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On

Files\\On testing machine learning programs

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On
No	Web of science	0.0311	14			
				1	S	07/02/2022 10:15

This paper reviews current existing testing practices for ML programs. First, we identify and explain challenges that should be addressed when testing ML programs. Next, we report existing solutions found in the literature for testing ML programs. Finally, we identify gaps in the literature related to the testing of ML programs and make recommendations of future research directions for the scientific community. We hope that this comprehensive review of software testing practices will help ML engineers identify the right approach to improve the reliability of their ML-based systems. We also hope that the research community will act on our proposed research directions to advance the state of the art of testing for ML programs.

				2	S	07/02/2022 10:17

In this paper, we survey existing testing practices that
have been proposed for ML programs, explaining the context in which they can be applied and their expected outcome. We also, identify gaps in the literature related to the testing of ML programs and suggest future research directions for the scientific community. This paper makes the following contributions:
• We present and explain challenges related to the testing of ML programs that use differentiable models.
• We provide a comprehensive review of current software testing practices for ML programs.
• We identify gaps in the literature related to the testing of ML programs and provide future research directions for the scientific

				3	S	07/02/2022 10:24

Approaches that aim to detect conceptual errors in ML models
Approaches in this category assume that the models
are implemented into programs without errors and focus on providing mechanisms to detect potential errors in the calibration of the models. These approaches can be divided in two groups: black-box and white-box approaches [9].

				4	S	07/02/2022 10:24

Black-box testing approaches for ML models. The common denominator to black-box testing approaches is the generation of adversarial data set that is used to test the ML models. These approaches leverage statistical analysis techniques to devise a multidimensional random process that can generate data with the same statistical characteristics as the input data of the model.

				5	S	07/02/2022 10:24

White-box testing approaches for ML models. Pei et al. proposed DeepXplore [15], the first white-box

				6	S	07/02/2022 10:25

Approaches that aim to detect errors in ML code implementations
Given the stochastic nature of most ML algorithms and
the absence of oracles, most existing testing techniques are inadequate for ML code implementations. As a consequence, the ML community have resorted to numerical testing, property-based testing

				7	S	07/02/2022 10:25

Numerical-based testing: Finite-difference techniques. Most machine learning algorithms are formulated as optimization problems that can be solved using gradientbased optimizers, such as gradient descent or L-BFGS (i.e., Limited-memory Broyden–Fletcher–Goldfarb–Shanno algorithm). The correctness of the objective function gradient that are computed with respect to the model parameters, is crucial.

				8	S	07/02/2022 10:26

a)Use of the centered formula. Instead of relying on the traditional gradient formula, Karpathy recommends using the centered formula from Equation 1 which is more precise. The taylor expansion of the numerator

				9	S	07/02/2022 10:26

b) Use of relative error for the comparison. As mentioned above, developers perform gradient checking by computing the difference between the numerical gradient f
n and the analytic gradient f a. This difference can be
seen as an absolute error and the aim of the gradient checking

				10	S	07/02/2022 10:26

d) Stick around active range of floating point. To train complex statistical models, one needs large amounts of data. So, it is common to opt for mini-batch stochastic gradient descent and to normalize the loss function over the batch. However, if the back-propagated gradient is very small, additional divisions by data inputs count will yield extremely smaller vales, which in turn can lead to numerical

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On
				11	S	07/02/2022 10:26

Property-based testing. Property-based testing is a technique that consists in inferring the properties of a computation using the theory and formulating invariants that should be satisfied by the code.

				12	S	07/02/2022 10:26

Metamorphic testing. Murphy et al. [32] introduced metamorphic testing to ML in 2008. They defined several Metamorphic Relationships (MRs) that can be classified into six categories (i.e., additive, multiplicative, permutative, invertive, inclusive, and exclusive).

				13	S	07/02/2022 10:26

Mutation testing. Ma et al.[36] proposed DeepMutation that adapts mutation testing [37] to DNN-based systems with the aim of evaluating the test data quality in terms of its capacity to detect faults in the programs. Mutation testing consists in injecting artificial faults (i.e., mutants) in a program under test and generating test cases to detect them.

				14	S	07/02/2022 10:27

Coverage-Guided Fuzzing. Odena and Goodfellow [38] developed a coverage-guided fuzzing framework specialized for testing neural networks. Coverage-guided fuzzing has been used in traditional software testing to find critical errors. For ML code, the fuzzing process consists of handling an input corpus that evolves through the execution of tests by applying random mutation operations on its contained data and keeping only interesting instances that allow triggering new program behavior.

Files\\Software Framework for Data Fault Injection to Test Machine Learning Systems

No	Web of science	0.0619	3			
				1	S	03/02/2022 15:36

Conceptual view of the system with examples of questions that the system can answer.
software framework, illustrated conceptually in Fig. 1, with the following goals:
– Easy and flexible modeling of the types of data faults the system is likely to encounter via a combination of predefined parameterizable fault models and new userdefined ones. Ideally, the set of predefined fault models should gradually grow as a result of the integration of new fault models for new purposes.
– Ability to work with different kinds of structured and unstructured data as well as with highly different ML models or systems.
– Parameterization of the data fault generation so that developers can study how sensitive their systems are to different kinds of faults and what are the thresholds of data problems when the system starts to lose its performance.
– Visualization of the results with different fault models and parameters.
– Bookkeeping to allow going back to the sources of problems.
– Embedding the fault injection and the visualization of the effects of faulty data to the development pipeline.
– Integration of data fault emulation to different development pipelines.
To satisfy these goals, we have created a generator frame-
work for emulating data problems, called dpEmu. The framework can generate faults in training or testing data in a controlled and documentable manner, and it enables emulating data problems in the use and training of ML systems as depicted in Fig. 2. The Runner routine introduces faults in a dataset, following the definitions set in the Fault generation tree. Then, the resulting data is preprocessed and run by ML models, which

				2	S	03/02/2022 15:36

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On
				3	S	03/02/2022 15:37

To develop robust and reliable ML systems we have created
dpEmu to encourage developers to evaluate how their models and systems work when system input data has faults. The system can be used for multiple purposes, such as investigating how a trained model or an entire system tolerate different kinds of faults in its input data; studying which model and hyperparameterization are the best when input data has certain kinds of faults or how alternative data cleaning approaches influence the operation of the resulting model; evaluating tradeoffs between model accuracy versus model robustness; and quantifying the accuracy difference when the model is trained with clean or faulty data. At present, dpEmu still is a prototype and as usual, it is not perfect in terms of functionality and usability. However, it already acts as demonstrator regarding how robustness and tolerance to data faults can be integrated into the development pipeline of ML systems. Popular ML libraries, such as Sklearn or TensorFlow, have
extensive collections of functions for evaluating the models as well as splitting the datasets for training and testing parts. However, none of these, seem to have built-in support for studying the model behavior in case of erroneous input data. DpEmu can be used together with these libraries to add one step before the actual training of the model. The addition of one more step, however, will increase the training effort a lot. In addition to the actual training, it is possible to have another training loop that searches for the best possible

Files\\TensorFI~ A Configurable Fault Injector for TensorFlow Applications

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On
No	Scopus	0.0216	3			
				1	S	03/02/2022 15:05

TensorFlow is a high-level dataflow framework for building ML applications and has become the most popular one in the recent past. ML applications are also being increasingly used in safety-critical systems such as self-driving cars and home robotics. Therefore, there is a compelling need to evaluate the resilience of ML applications built using frameworks such as TensorFlow. In this paper, we build a high-level fault injection framework for TensorFlow called TensorFI for evaluating the resilience of ML applications. TensorFI is flexible, easy to use, and portable. It also allows ML application programmers to explore the effects of different parameters and algorithms on error

				2	S	03/02/2022 15:07

In this paper, we build a fault injector for ML applications
written using specialized frameworks. Because TensorFlow is the most widely used, publicly available software framework for writing ML applications today, we only support TensorFlow and we call our injector TensorFI. TensorFI has three main features. First, it does not rely on the internal implementation of TensorFlow, aiding its portability to different platforms and TensorFlow versions. Second, it requires minimal modifications for programmers to make to their applications and is hence easy to use. Third, it allows programmers to configure the injection process through an external interface without modifying the application (flexible).

				3	S	03/02/2022 15:07

E. Implementation TensorFI supports the following features: • Launching multiple FI runs with support for comparing each FI result with the golden run
• Launching multiple FI runs in parallel (multi-threading) • Support for visualizing the modified TensorFlow graphs • Ability to specify fault type etc. in a configuration file • Automated logging of fault injection runs • Support for statistics collection and analysis

Nodes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Building ML Systems and applications\Architecture ML system

PDF

Files\\AI lifecycle models need to be revised

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On
No	Google Scholar	0.0252	13			
				1	S	10/02/2022 11:49

We have found that the following stages have been overlooked by previous lifecycle models: data collection, feasibility study, documentation, model monitoring,and model risk assessment.Our work shows that the real challenges of applying Machine Learning go much beyond sophisticated learning algorithms – more focus is needed on the entire lifecycle. In particular, regardless of the existing development tools for Machine Learning, we observe that they are still not meeting the particularities of this field.

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On
				2	S	10/02/2022 11:52

MachineLearning projects start witha problem statement which is used to discuss
whether a MachineLearning solution is necessary.This step requires high engagement from problem domain experts.

				3	S	10/02/2022 11:52

Requirements are not alwaysdefined beforehand. DataandModel requirements
become more clear while working withan initial model. Requirements relatedtotraceability,interpretability, andexplainabilityare typically defined a theorganizational level.

				4	S	10/02/2022 11:52

Collecting, understanding, andpreparing dataare the most time-consuming stages
ofMachineLearning projects. There is ameticulousdataaccess control that, despite being quintessential, sets major obstacles inunderstanding thedataand performing exploratory analyses. Practitioners emphasize, dataunderstanding implies being able to communicate it to other stakeholders. Finally,the differences between development andproduction environments pose challenges for

				5	S	10/02/2022 11:52

Thechallenges in Modeling summarize as follows: 1) thelatest MachineLearning
technologies are not alwayseligible for use;2)baseline models are essential artifacts for model development; 3) teams keep track of all experiments, which often revolves around keeping a customized spreadsheet; and4)defining performance metrics is problem-specific, posing a challengeto thedefinition ofstandards attheorganizational level.

				6	S	10/02/2022 11:52

Documentation is a first-class artifact for regulatory compliance, knowledge
transfer, andreproducibility.Hence, a peer-review process is in place to ensure documentation quality.

				7	S	10/02/2022 11:52

Although Model Risk Assessmentis not new to thefintech industry,MachineLearn-
ing is requiring a revised approach.Currently,developers endure considerable efforts to create therequired documentation.

				8	S	10/02/2022 11:52

There are deploymentpatternsinwhicha separate team needs to reimplementthe model to meet production settings.

				9	S	10/02/2022 11:53

More automation is needed for model monitoring.Teams havecreated their own
automation tools, but making themavailable to other teamsrequires unfeasible efforts thatdo not meet their priorities.

				10	S	10/02/2022 11:53

Although practitioners are eager to learnautomated testing practices, this is not
part oftheir skillset. Hence, projects are struggling to adopt unit andintegration testing strategies.

				11	S	10/02/2022 11:53

All projects must goover a feasibility study in their early stages. Until then, projects
do not fit thetypicalsprint-based agile planning.Anagile approachhelps practitioners prioritize tasks andengagestakeholders.

				12	S	10/02/2022 11:53

There is practical valueonhaving a strong backgroundon bothSoftware Engi-
neering and DataScience. Education should put more focuson theprocess insteadof model-training techniques.

				13	S	10/02/2022 11:54

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On

Files\\An End-to-End Framework for Productive Use of Machine Learning in Software Analytics and Business Intelligence Solutions

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On
No	Google Scholar	0.0941	11			
				1	S	09/02/2022 13:22

Nowadays, machine learning (ML) is an integral component in a wide range of areas, including software analytics (SA) and business intelligence (BI). As a result, the interest in custom ML-based software analytics and business intelligence solutions is rising. In practice, however, such solutions often get stuck in a prototypical stage because setting up an infrastructure for deployment and maintenance is considered complex and time-consuming. For this reason, we aim at structuring the entire process and making it more transparent by deriving an end-to-end framework from existing literature for building and deploying ML-based software analytics and business

				2	S	09/02/2022 13:28

there is often a need for customized software analytics or business intelligence (SA/BI) solutions that leverage the full potential of modern machine learning (ML) techniques. However, as such solutions are used as internal systems for monitoring or decision-making, these are often not perceived as something of direct customer value by managers. This results in a lack of priority, time and, resources assigned to setup and maintain ML-based SA/BI solutions [15]. In addition to that, the effort of going beyond a prototypical analysis and deploying it to and maintaining it in production is perceived as extremely high [15,30]. Paired with a lack of expertise in this domain, which is often the case if the actual product is not related to ML [6], custom ML-based SA/BI solutions are rarely deployed in production [15]. Nevertheless, this is considered crucial in order to continuously gain valuable insights and use it for actual decision making [21].

				3	S	09/02/2022 13:30

Data Management and Processing
The most important prerequisite for training accurate ML models is providing high-quality training data [26,29]. At the same time, assembling high-quality data sets, and engineering and selecting appropriate features based on it, is very time-consuming and requires a vast amount of effort and resources [14]. As a result, we investigate the common activities (see Table 1) in data man-
agement and data processing required for a successful application in machine learning systems as well as the challenges (see Table 2) that come with these activities. The identified activities can be grouped into six overarching categories: 1) Data preparation; 2) data cleaning; 3) data validation; 4) data evaluation; 5) data serving; and 6) extract, transform, and load (ETL) tasks. During the data preparation, raw input data is examined for suitable features
before being transformed (e.g. aggregations of one or more raw input data fields) into training data [4,5,14,21,26,27]. Next, the data is cleaned by filtering out uncorrelated data [10,26], specifying quality rules, detecting errors, inconsistencies and anomalies [4,8,19], and fixing these errors [8,19,26,36]. To guarantee a successful preparation and cleaning of the data, each batch of
data needs to be validated based on its properties [4,5,26,27,29,36] and potential

				4	S	09/02/2022 13:30

dependencies [26], deviations [5,26], or impact of features on model accuracy or performance [14,26] need to be identified. Once a model is trained, the goal of data evaluation is to evaluate the choice
and encoding of the data based on the results produced by a model trained on the data, for instance by performing sanity checks [14,26]. After a suitable solution was found, the newly emerging input data needs to be transformed to so-called serving data which is processible by the model [4,26]. This usually involves the same transformation steps as required for the training data. After the serving data was successfully processed by the model, it is channeled back as training data for future iterations [26].

				5	S	09/02/2022 13:31
				6	S	09/02/2022 13:32
				7	S	09/02/2022 13:32
				8	S	09/02/2022 13:33
				9	S	09/02/2022 13:34

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On
				10	S	09/02/2022 13:34
				11	S	09/02/2022 13:34

Files\\Applying AI in Practice~ Key Challenges and Lessons Learned

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On
No	Scopus	0.0183	4			
				1	S	08/02/2022 13:54

In particular, data-driven AI methods such as DNNs allow data to shape models and software systems that operate them. System engineering of AI-driven software therefore faces novel challenges at all stages of the system lifecycle [51]: – Key Challenge 1: AI intrinsic challenges due to peculiarities or shortcomings of today's AI methods; in particular, current data-driven AI is characterized by: • data challenge in terms of quality assurance and procurement; • challenge to integrate expert knowledge and models; • model integrity and reproducibility challenge due to unstable performance profiles triggered by small variations in the implementation or input data (adversarial noise);
– Key Challenge 2: Challenges in the process of AI system engineering ranging from requirements analysis and specification to deployment including • testing, debugging and documentation challenges; • challenge to consider the constraints of target platforms at design time; • certification and regulation challenges resulting from highly regulated target domains such as in a bio-medical laboratory setting;
– Key Challenge 3: Interpretability and trust challenge in the operational environment, in particular • trust challenge in terms of lack of interpretability and transparency by opaque models;

				2	S	08/02/2022 13:54

AI Intrinsic Challenges
There are peculiarities of deep learning methods that affect the correct interpretation of the system's output and the transparency of the system's configuration.
Lack of Uniqueness of Internal Configuration: First of all, in contrast to traditional engineering, there is a lack of uniqueness of internal configuration causing difficulties in model comparison. Systems based on

				3	S	08/02/2022 13:54

AI System Engineering Challenges
In a data-driven AI systems there are two equally consequential components: software code and data. However, some input data are

				4	S	08/02/2022 13:54

Interpretability and Trust Challenge In contrast to traditional computing, AI can now perform tasks that previously
only humans were able to do. As such it contains the possibility to revolutionize every aspect of our society.

Files\\Bighead~ A Framework-Agnostic, End-to-End Machine Learning Platform

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On
No	IEEE	0.0549	4			
				1	S	08/02/2022 12:52

With the increasing need to build systems and
products powered by machine learning inside organizations, it is critical to have a platform that provides machine learning practitioners with a unified environment to easily prototype, deploy, and maintain their models at scale. However, due to the diversity of machine learning libraries, the inconsistency between environments, and various scalability requirement, there is no existing work to date that addresses all of these challenges.

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On
				2	S	08/02/2022 12:54

Despite rapid developments in the field, there still lacks
a framework-agnostic, end-to-end machine learning platform, and existing solutions do not satisfy the needs of machine learning practitioners. First of all, many platforms lack advanced feature engineering capability, leaving many challenges unsolved in a stage in model development where many practitioners spend the majority of their time [1]. For example, it is crucial to have correct values for the features that correspond to the timestamp of the labels. This process, called temporal joins, prevents the situation of data leakage [2], that is, features incorrectly containing information on the labels because the former were observed after the latter. Another challenge is that, for features that are generated and consumed in real time, we need a framework that can process, aggregate, and join both offline and online data sources. This is not a trivial problem since aggregations and temporal joins need to be properly modeled in a principled way. Moreover, existing platforms typically focus on supporting
only one model framework, often leading to tight coupling between the modeling layer and the infrastructure layer. This limits the options for practitioners when they build models, and can prevent cutting-edge algorithms and techniques from being explored and adopted. It also creates a lock-in with certain frameworks and makes migrations difficult when these frameworks evolve or get deprecated. Apart from the drawbacks of existing systems, we have identified the following four major overarching challenges when building a well designed machine learning platform. First, it is common for model developers to spend a non-
trivial amount of effort to iterate on models and take them to production. Cleaning up the code, writing applications to serve the model, and thoroughly testing changes are frequent tasks. In some cases, developers even have to re-implement

				3	S	08/02/2022 12:55

Second, the domain of machine learning is highly hetero-
geneous and ever-changing. Models using certain algorithms are typically built on structured data, and state-of-the-art deep learning models can leverage unstructured data such as texts, images, and videos, each of which require unique processing. Meanwhile, algorithms, frameworks, and platforms are constantly being released and updated. New compute resources such as GPUs and TPUs are increasingly required. For such a platform to be useful, it needs to be versatile by supporting major frameworks and various compute resources, being flexible to accommodate frequent changes, and being extensible to allow future additions. To achieve these goals, the platform needs to decouple infrastructure from the model frameworks and provide proper abstractions. Third, models are moved across a diverse set of environments throughout their lifecycle. These environments can differ in numerous aspects, such as hardware, operating systems, versions of software dependencies, and sources of data. For example, the production environment is often vastly different from the prototyping environment. Data used for offline training often comes from a different source from online inference. Consequently, data produced by the model in production can be inconsistent with that produced during prototyping, leading to undesired situations such as incorrect results. It is therefore important to guarantee that the models are developed and productionized in a consistent setting and produce consistent results. Fourth, the scales of the datasets, throughput, latency requirements, etc. all vary drastically from model to model, and can fluctuate greatly over time. A modern convolution neural network can easily consume thousands of times more resources than a simple regression model. A fraud detection model may require sub-second latency, whereas a sales forecasting model may only need to run once per month. While having as much computing power as possible is one way to solve the scaling problem, cost adds constraints on how many resources can be deployed at a time. The ability to scale horizontally and elastically in response to the change of the workload is thus critical to the stability, reliability, and cost effectiveness of the platform.

				4	S	08/02/2022 12:56

We found that these platforms do not meet the need by the machine learning community for a framework-agnostic, endto-end platform, for several reasons. First, many of them do not cover the end-to-end workflow. In particular, an important feature that most platforms lack is the integration with feature engineering and management, which is considered by some to be the most crucial part of machine learning [1]. As mentioned in Section I, there exist many challenging problems pertaining to this stage that a platform needs to solve. Second, existing platforms focus on the support of one machine learning framework, thus not giving first-class support for or even precluding the use of others. Moreover, most of the frameworks are not designed in a flexible way, and substantial work would be required for customized features, such as integration with a particular organization's data warehouse, or enforcement of data privacy policies. Lastly, some platforms are proprietary, and while they might
have a more complete coverage for the workflow or popular frameworks, they cannot be leveraged by other organizations. For the above reasons, we decided to build Bighead on our
own, while leveraging existing open source technologies as much as possible, such as Apache Spark [19], Apache Flink [20], Apache Airflow [21], and Kubernetes [22]. Rather than stitching separately developed components together, Bighead

Files\\Large-scale machine learning systems in real-world industrial settings~ A review of challenges and solutions

No	Web of science	0.0101	1			
				1	S	11/02/2022 14:22

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On

Files\\Overton~ A Data System for Monitoring and Improving Machine-Learned Products

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On
No	Google Scholar	0.0117	1			
				1	S	24/02/2022 09:54

In the life cycle of many production machine-learning applications, maintaining and improving deployed models is the dominant factor in their total cost and effectiveness–much greater than the cost of de novo model construction. Yet, there is little tooling for model life-cycle support. For such applications, a key task for supporting engineers is to improve and maintain the quality in the face of changes to the input distribution and new production features. This work describes a new style of data management system called Overton that provides abstractions to support the model life cycle by helping build models, manage supervision, and monitor application quality.1 Overton is used in both near-real-time and backend production applications. However, for concreteness, our running
example is a product that answers factoid queries, such as "how tall is the president of the united states?" In our experience, the engineers who maintain such machine learning products face several challenges on which they spend the bulk of their time.

Nodes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Building ML Systems and applications\Architecture ML system\Cloud_based ML

PDF

Files\\ThunderML~ A Toolkit for Enabling AI~ML Models on Cloud for Industry 4.0

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On
No	Google Scholar	0.0185	1			
				1	S	03/02/2022 11:15

Challenges of Using Existing Cloud-Based AI Platforms
While cloud-based AI platforms have done much to facilitate adoption of AI by alleviating many of the infrastructure provisioning and maintenance challenges associated with on-premises enterprise AI initiatives, they have not done enough to abstract away some of the complexity of running AI workflows in vendor agnostic ways. Current platforms expect practitioners to know a given vendor's means and methods of interacting with the computing resources without consideration given to providing a common programming model that makes the job of an AI practitioner easier. Cloud-based AI environments, by their very nature, push users towards batch training modes to facilitate data center resource management via a queued execution model. Such batch training modes are problematic for many data scientists who wish to see errors or results in real or near real time in order to make their modeling workflow more efficient.1 Another issue is that cloud-offerings typically approach AI from either a
black-box perspective which offers users simplicity at the cost of flexibility or through a more complex runtime environment that requires users maintain code artifacts that often have nothing to do with the actual AI tasks at hand2.Even with a diverse set of offerings in the market, we feel a gap remains for the AI practitioner community. Cloud AI offerings should be easy to learn and use and provide the right level of complexity and flexibility AI practitioners need.

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On

Nodes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Building ML Systems and applications\Architecture ML system\ML pipeline jungles

PDF

Files\\On the Co-evolution of ML Pipelines and Source Code - Empirical Study of DVC Projects

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On
No	Web of science	0.0047	1			
				1	S	04/02/2022 22:57

The growing popularity of machine learn-
ing (ML) applications has led to the introduction of software engineering tools such as Data Versioning Control (DVC), MLFlow and Pachyderm that enable versioning ML data, models, pipelines and model evaluation metrics. Since these versioned ML artifacts need to be synchronized not only with each other, but also with the source and test code of the software applications into which the models are integrated, prior findings on co-evolution and coupling between software artifacts might need to be revisited.

Nodes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Building ML Systems and applications\AutoML

PDF

Files\\A Meta Learning Approach for Automating Model Selection in Big Data Environments using Microservice and Container Virtualization Technologies

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On
No	ACM Digital library	0.0283	3			
				1	S	25/02/2022 10:07

For a given speci c machine learning task, very often several machine learning algorithms and their right con gurations are tested in a trial-and-error approach, until an adequate solution is found. This wastes human resources for constructing multiple models, requires a data analytics expert and is time-consuming, since a variety of learning algorithms are proposed in literature and the non-expert users do not know which one to use in order to obtain good performance results. Meta learning addresses these problems and supports non-expert users by recommending a promising learning algorithm based on meta features computed from a given dataset. In the present paper, a new generic microservice-based framework for realizing the concept ofmeta learning in Big Data environments is introduced. This framework makes use of a powerful Big Data software stack, container visualization, modern web technologies and a microservice architecture for a fully manageable and highly scalable solution. In this demonstration and for evaluation purpose, time series model selection is taken into account. The performance and usability of the new framework is evaluated on state-of-the-art machine learning algorithms for time series forecasting: it is shown that the proposed microservice-based meta learning framework introduces an excellent performance in assigning the adequate forecasting model for the chosen time series datasets.

Reference Number	Coded By Initials	Modified On
2	S	11/02/2022 12:42

Meta learning solves the problem of automated model selection
by formulating it as a supervised learning task incorporating training and testing phases. The main task of the training is that an algorithm –referred to meta learner– learns the mapping between available learning algorithms and measurable properties –referred to meta features– of the task itself. To this end, a set of meta examples is needed. Each example is tagged by predictors and labels. The predictors correspond to the meta features extracted to describe the datasets. The labels indicate the most appropriate algorithms. In the testing phase, given a new dataset, the meta features will be extracted to be used by the trained meta learner for suggesting the most

Reference Number	Coded By Initials	Modified On
3	S	11/02/2022 12:42

Building a ML model by nonexpert users is a complex, time consuming and error prone process. According to the no-free-lunch theorem [31], no single learning algorithm has always the lowest performance error on a broad problem domain. As a result, for a given usage scenario, a dedicated learning algorithm must be selected. The selection process is de ned as an Algorithm Selection Problem (ASP) [18],

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On

Files\\Auto-Keras~ An Efficient Neural Architecture Search System

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On
No	Google Scholar	0.0170	2			
				1	S	08/02/2022 13:43

Initial efforts have been devoted to making use of network mor-
phism in neural architecture search [7, 13]. It is a technique to morph the architecture of a neural network but keep its functionality [10, 45]. Therefore, we are able to modify a trained neural network into a new architecture using the network morphism operations, e.g., inserting a layer or adding a skip-connection. Only a few more epochs are required to further train the new architecture towards better performance. Using network morphism would reduce the average training time ¯t in neural architecture search. The most important problem to solve for network morphism-based NAS methods is the selection of operations, which is to select an operation from the network morphism operation set to morph an existing architecture to a new one. The network morphism-based NAS methods are not efficient enough. They either require a large number of training examples [7], or inefficient in exploring the large search space [13]. How to perform efficient neural architecture search with network morphism remains a challenging problem.

				2	S	08/02/2022 13:43

As we know, Bayesian optimization [40] has been widely adopted
to efficiently explore black-box functions for global optimization, whose observations are expensive to obtain. For example, it has been used in hyperparameter tuning for machine learning models [3, 15, 21, 24, 40, 44], in which Bayesian optimization searches among different combinations of hyperparameters. During the search, each evaluation of a combination of hyperparameters involves an expensive process of training and testing the machine learning model, which is very similar to the NAS problem. The unique properties of Bayesian optimization motivate us to explore its capability in guiding the network morphism to reduce the number of trained neural networks n to make the search more efficient. It is non-trivial to design a Bayesian optimization method for
network morphism-based NAS due to the following challenges. First, the underlying Gaussian process (GP) is traditionally used for learning probability distribution of functions in Euclidean space. To update the Bayesian optimization model with observations, the underlying GP is to be trained with the searched architectures and their performances. However, the neural network architectures are not in Euclidean space and hard to parameterize into a fixed-length vector.

Files\\Autonomic machine learning platform

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On
No	Google Scholar	0.0151	3			
				1	S	11/02/2022 14:50

Autonomic machine learning level In this section, first we define autonomic machine learning based on
minimizing expert intervention. We also define the autonomic levels based on the development factors of the machine learning process. 3.1. Autonomic machine learning Similar to the concept of autonomic computing, which refers to a
computing framework to manage, configure, and optimize the assets of all systems while minimizing expert intervention (Autonomic Computing Strategy Perspectives, 2018), autonomic machine learning refers to autonomic machine learning with minimal expert intervention. Therefore, autonomic machine learning can be defined as the selection of an appropriate machine learning model and algorithm to achieve the desired result while autonomously detecting the

				2	S	11/02/2022 14:50

Autonomic levels In order to develop machine learning applications which work by
learning target data, machine learning experts generally undertake the processes of selecting the attributes of the input data, tuning the hyperparameters, and selecting the machine learning technique and learning task. If we categorize these processes into the development steps of machine learning and if the process of each step can be performed without expert intervention, each step can then be defined as a level of autonomic machine learning. Therefore, we define five levels of autonomic machine learning referring to as the degree of expert intervention based on the development steps of machine learning. These are shown in Table 1.

				3	S	11/02/2022 14:51

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On

Files\\AutoTrain~ An Efficient Auto-training System for Small-scale Image Classification

No	IEEE	0.0072	1			
				1	S	08/02/2022 13:13

Machine learning has become the most promising
research field. However, the involved models usually require complex, tedious and expensive manual intervention. The automated machine learning technology plays a significant role in mitigating this issue. However, the current studies ignore the importance of automation in data preprocessing.

Files\\Task-Specific Automation in Deep Learning Processes

No	Web of science	0.0245	1			
				1	S	25/02/2022 11:13

In this paper, we looked on the challenges of AI system development from a SE point of view. These challenges lead to new ML processes automated in general purpose ML pipelines. These pipelines still lack the ability to support the huge diversity of task and technology specific requirements ofML solutions. Automatic ML ('learn how to learn') aiming at full end-to-end pipeline synthesis is promising but still not mature enough for large scale application in industry projects. We argued task and technology specific automation taking advantage from both approaches are the next steps towards better ML pipelines. As an examplewe presented the ALOHA tool flow automating the steps ofalgorithm
selection, application partitioning and mapping and deployment on target hardware. We presented the evaluation method, based on real world industry relevant use cases. Although the ALOHA project is still ongoing, examples based on already implemented components strongly suggest the capability of the ALOHA tool flow to provide designs optimized for specific hardware platforms in a matter of days,

Files\\The Machine Learning Bazaar~ Harnessing the ML Ecosystem for Effective System Development

No	Google Scholar	0.0056	1			
				1	S	25/02/2022 10:00

. To address these problems, we introduce the Machine Learning Bazaar, a new framework for developing machine learning and automated machine learning software systems. First, we introduce ML primitives, a unified API and specification for data processing and ML components from different software libraries. Next, we compose primitives into usable ML pipelines, abstracting away glue code, data flow, and data storage. We further pair these pipelines with a hierarchy ofAutoML strategies — Bayesian optimization and bandit learning. We use these components to create a general-purpose, multi-task, end-to-end AutoML system that provides solutions to a variety ofdata modalities (image, text, graph, tabular, relational, etc.) and problem types (classification, regression, anomaly detection, graph matching, etc.). We demonstrate 5 real-world use cases and 2 case studies of our approach.

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On

Nodes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Building ML Systems and applications\Machine learning System Quality

PDF

Files\\Cats are not fish~ deep learning testing calls for out-of-distribution awareness

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On
No	ACM Digital library	0.0175	3			
				1	S	08/02/2022 12:35

As Deep Learning (DL) is continuously adopted in many industrial applications, its quality and reliability start to raise concerns. Similar to the traditional software development process, testing the DL software to uncover its defects at an early stage is an effective way to reduce risks after deployment. According to the fundamental assumption of deep learning, the DL software does not provide statistical guarantee and has limited capability in handling data that falls outside of its learned distribution, i.e., out-of-distribution (OOD) data.

				2	S	08/02/2022 12:38

However, different from traditional software whose decision
logic is mostly programmed by the developer, deep learning adopts a data-driven programming paradigm. In particular, the major tasks of a DL developer are preparing the training data, labeling the data, programming the architecture of the deep neural network (DNN), and specifying the training configuration. All the decision logic is automatically learned during the runtime training phase and encoded in the obtained DNN (e.g., by weights, bias, and their combinations). Due to the differences of programming paradigm, the logic encoding format, and the tasks that a DNN is often developed for (e.g., image recognition), testing techniques for traditional software cannot be directly applied and new testing techniques are needed for DNNs. While some recent progress has been made in proposing novel testing criteria [17, 25, 33, 35] and test generation techniques for quality assurance of DNNs [8, 33, 35, 43, 48, 55, 58], it still lacks interpretation and understanding on the detected errors by such techniques and their impact. For example, it is not clear whether errors are indeed caused by missing training data or insufficient training, etc. The fundamental assumption of deep learning is that

				3	S	08/02/2022 12:40

To summarize, this paper makes the following contributions:
• We perform a large scale empirical study on how deep learning testing affects the data distribution of the generated test cases; and how distribution aware testing influences DNN model robustness.
• Our study identifies the impact of mutation operators and coverage criteria on the distribution of the generated test cases. We find that image rotation, contrast and brightness tend to generate more ID data while image blur is more likely to generate OOD data. In terms of the coverage criteria, NBC and SNAC facilitate to generate more OOD data than others.
• We demonstrate the effectiveness of distribution aware retrain-
ing, outperforming the state-of-the-art by up to 21.5%. Based on our results, we provide guidelines on distribution-aware error selection for robustness enhancement, by studying the effect of root cause of ID and OOD errors.

Files\\How Teams Communicate about the Quality of ML Models~ A Case Study at an International Technology Company

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On
No	ACM Digital library	0.0620	14			
				1	S	07/02/2022 16:27

Prior studies have explored how ML is affecting development team roles beyond data scientists, including user experience designers, program managers, developers and operations engineers. However, there has been little investigation ofhowteam members in different roles on the team communicate about ML, in particular about the quality ofmodels. We use the general term quality to look beyond technical issues ofmodel evaluation, such as accuracy and overfitting, to any issue affecting whether a model is suitable for use, including ethical, engineering, operations, and legal considerations. What challenges do teams face in discussing the quality ofML models? What work practices mitigate those challenges? To

				2	S	07/02/2022 16:27

This emergence of ML also means that software teams increasingly need to communicate
about ML models and their quality. Here, we use the general term quality to encompass not only technical issues about ML model evaluation such as accuracy, but also any issue affecting whether a model is suitable for use, including ethical, engineering, operations, and legal considerations. Because ML is involved in many aspects of software development—from UX, to engineering and operations, to management—this communication spans many roles on the team [7, 37]. Prior research explored how data scientists ensure high confidence in their analysis results [24, 36, 37, 57]. However, concerns about ML quality are not limited to issues related to the role of data scientists. Other roles on the team, including UX designers, program managers (PMs), developers, operations engineers, and product managers, also need to communicate about ML models and their quality, to support coordination and decision making. What are the challenges software teams face when communicating about ML models and their quality? What practices and tools can be introduced to

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On
				3	S	07/02/2022 16:29

Challenges in Communicating ML Models In this section, we discuss the main challenges and best practices around communicating ML models within teams. In the surveys and interviews, participants were asked two questions about challenges: One was aimed at ML experts who develop the models, asking them directly about challenges they face when discussing ML models with non-experts. The other question was directed toward non-ML developers working on ML projects and the challenges they face within their teams when those ML models are discussed. Through another affinity diagram exercise, we iterated over the reported open-ended questions, we identified emerging patterns. We share the following challenges and best practices to inspire future work and design direction for new solutions.
5.1.1 Mismatch between the discussion ofuser-need-driven and model-driven performance. One of the common problems that has been observed in a variety of ways is that when discussing ML features in a system, data scientists tend to be more model driven as opposed to user-need driven (e.g. they discuss the performance of a model in terms of accuracy or recall, but not in terms of the overall task). This mostly affects those ML model developers who are building customer facing features ofmodels. For example, a software engineer mentioned the challenge he faces when discussing the model as follows: “Our team focuses on building tools for 3rd party developers, so our work is generally focused on models as collections/generally. It’s often hard to discuss the models when we’re focused on how they perform on a standard task (ImageNet, CoCo) but customers care about their specific problem space”. Another participant who is a UX designer working on customer facing ML based projects explains the situation with the following metaphor:“Trying to navigate the ’cart before the horse’ discussion because it can be a sensitive topic. Cart before the horse = conducting research, creating and training models before having a sense ofwhat real customer problem you’re trying to solve. Can lead to sometimes feeling like we’re shoe-horning ML

				4	S	07/02/2022 16:29

5.1.2 Struggle in problem formulation. Another emerging pattern in our observations from the survey and interviews is due to the lack of knowledge regarding the capabilities and potentials of ML. The problem is divided into two groups: One occurs at the beginning of the ML development workflow due to issues in problem formulation as a result of not knowing what to expect from ML, and the other occurs during validation . For example, a data scientist clearly described the challenge he faces regarding his audience expectations: “Many non-technical people assume that ML can do anything and that it will tell them what they need to know. But ML is only good when you have 1) good questions, and 2) good data (with good labels!).

				5	S	07/02/2022 16:29

5.1.3 The need for education before communication . Not only is the lack ofML knowledge challenging at the start of the workflow during problem formulation, but it also requires extra effort from the model developer to educate and raise awareness in their audience. As one data scientist explained, “We usually need to put in extra work to achieve the baseline level ofknowledge before we can discuss the actual mode/feature”. Another data scientist mentioned how this mismatch in the level of knowledge forces her to hide some details to avoid misunderstanding. In her words, the challenge is “[h]ow to convert technical terminology into daily words. Sometimes, in order to explain clearly, I have to ignore some exceptions/details to avoid confusion”.

				6	S	07/02/2022 16:29

5.1.4 Conflicting documentation and standards. . One of the interesting patterns in the results is how a lack of standards and documentation that are universal across team members is a hurdle in the communication process because no common language is available. One data scientist explained, “The biggest challenge I’ve run into on our team is folks having a common language to use when discussing models in terms offunctionality.” He then followed up with how it is less of a challenge due to his team process “Fortunately our APIconsumes the ONNXmodel format which has helped improve these conversations.”. Another data scientist mentioned, “There is no standard checklist for checking the model’s accuracy. When you go and talk to different data scientists, even seniors they have their own mind map, so going through the model with different people is different”.

				7	S	07/02/2022 16:30

5.1.5 Failure to see ‘The Big Picture’. Another challenge around communicating models arises from the fact that an ML model is part of an ecosystem in which conversations are bidirectional. The challenge arises out of the fact that some data scientists are not aware of the deployment and engineering practices. For example one PM mentioned: “Modelers don’t always know what it takes to take a model to production”. An SE confirmed this by stating: “Scoping work can be difficult; it’s not always easy to know how many models we’ll need to try and how long we’ll need to iterate”.

				8	S	07/02/2022 16:30

5.1.6 Struggle to explain and understand common model metrics in context. Variations in metrics and their subjectivity raises a challenge in discussions within teams. For example one a data scientist mentioned, “Most people do not understand that it’s difficult to compare metrics across models - sometimes an AUC of0.9 is good enough, sometimes it is not”. Another senior data scientist mentioned something along those lines: “Sometimes a slightly low accuracy does not ruin the UX, it could be ignored or maybe it does not have impact. It is hard to convey that to our stakeholders”. Another issue related to communicating meaningful metrics is how they are being presented and discussed, because some data scientists choose to bias their presentation toward those metrics that work. As a data scientist expressed: “Not everyone knows what metrics really mean. Some people just blindly follow the metrics.

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On
				9	S	07/02/2022 16:30

5.1.7 Struggle to explain and interpret model behavior. There is often a difficulty in “explaining” a model’s behavior, (i.e., When the model works well, why? When the model is not working as

				10	S	07/02/2022 16:30

5.1.8 Intimidation by the perceived complexity, or is it too much trust? An obvious pattern in our interviews with data scientists is the issue of trust in the sense that a model consumer often trusts that the model builder knows their job; therefore, there is not as much discussion around the model as there might need to be. As one UX designer mentioned, “I have no formal education related to ML models. So basically, I just rely on the adhoc understanding ofthe model developer”. A PM also mentioned that, “stakeholders don’t care about model metrics they just want to know ifthe model is good or not, Usually I just share one number (aggregate measure like accuracy) and no one ever asked for details. They just want to see data related to the requirements”.

				11	S	07/02/2022 16:30

.3.3 Workflows. Sharing the process that led to the solution is also crucial in promoting confidence in the quality of the solution. Some details about ML models seem to be important to the audience, such as the reason for choosing a specific class ofmodels or how the parameters were tuned and so on. For instance, a data scientist mentioned what to discuss around ML models: “Problem being solved and why we picked the ML model? Your thought process behind picking the model”. A software engineer also suggested a more engaging discussion in which the results and the journey to the solution matter to non-ML developers “show how you’ve arrived at the results and how tweaking certain parameters can impact the model output

				12	S	07/02/2022 16:30

5.4.1 Presentation tools. Looking at the results of Table 3, we see some practices and tools that are currently used to present models’ quality to elicit launching decisions or any of the other goals shown in Table 4. PowerPoint and OneNote were popular in communicating models, which presents a potential design opportunity.

				13	S	07/02/2022 16:31

5.4.3 Quality shines with context and visuals. In presenting models, having the right form of presentation that relates the model to a context is a helpful exercise. As one PM mentioned regarding his approach when data scientists are presenting their models to him: “How explicable is this model? it helps to have a nice story around the model, what’s the story and how will it perform in the wild? the person has to [tell] a story.”.

				14	S	07/02/2022 16:31

Our results were elicited from observing the communication process through our interviews and surveys. Our questions were inspired by Lasswell’s model of communication [25] to provide a comprehensive analysis of the communication process. During our interviews, participants shared artifacts they used to discuss MLmodels (e.g., emails sent to other team members, presentations, projecting their work, etc.), which helped us in the elicitation process. However, a longer ethnographic shadowing would be beneficial in adding to the challenges and best practices we have identified. Our recruited interview and survey participants were all employees of a software company with large multidisciplinary teams. Most of our participants had some exposure to ML because the company is actively advertising events and courses to learn ML. Therefore, some of our observations might not generalize to less experienced teams. Furthermore, the total number of data scientists and software engineers was higher than the numbers of any other user group reflected in the software company population. We tried to randomize the sample by randomizing the questions regarding communication to

Files\\On misbehaviour and fault tolerance in machine learning systems

No	Web of science	0.0376	7			
				1	S	07/02/2022 10:42

We studied software designs that aim at introducing fault tolerance in ML systems so that possible problems in ML components of the systems can be avoided. The research was conducted as a case study, and its data was collected through five semi-structured interviews with experienced software architects.

				2	S	07/02/2022 10:42

Architectural designs have been suggested to protect the systems from hardware failures and malicious attacks (e.g. [6]), but there is little emphasis on architectural software design to answer the inherent unpredictability and uncertainty of the utilized ML itself. One step towards achieving dependability is fault tolerance.
Traditionally, software faults have been seen as the results of design errors [7]. However, due to their statistical, data-driven nature, ML systems can be seen as inherently faulty not by design, but by paradigm. Thus, unpredictable errors will emerge from deployed ML systems that cannot be captured by traditional fault-tolerance models. The question remains of how to build ML systems that detect these errors and prevent them from propagating.

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On
				3	S	07/02/2022 10:44

The results show that there is much to desire in the dependability of ML systems. Some patterns for fault tolerance are used in practice, but the developers and buyers often lack knowledge and frameworks to apply them. Thus, the role of fault tolerance is – at least today – very limited and vague in practice. This relative immaturity is not limited to fault tolerance, however, but also other phases of managing the life-cycle of ML systems. To our knowledge, this is the first attempt to gather information about fault tolerance for ML systems in one place, thus forming the basis for further research. Practitioners can use the gathered information to design more dependable ML systems.

				4	S	07/02/2022 10:44

Dependability, faults, and ML systems
System dependability means a system's trustworthiness [2]. Dependability is assessed by evaluating a system's reliability, availability, and maintainability. Sometimes additional quality characteristics, such as safety and integrity, are applied [10]. In other words, a dependable system – at the very least – delivers correct service consistently, does not suffer from long periods of down-time, and is easily corrected and altered. Threats to dependability originate from failures, errors, and
faults [10]. Failures are deviations from the desired service. Failures result from propagating errors, i.e., incorrect functioning of the system. Errors are caused by faults that are defects in system's components (software or hardware), activated by given inputs in a given

				5	S	07/02/2022 10:44

Two means of diminishing threats are fault prevention and fault tolerance.

				6	S	07/02/2022 10:48

Key findings are:
• ML system can provide poor results if the inputs are of poor quality, the input-output-pairs do not match, or the input distribution drifts. This can be caused by a buggy model, faulty deployment, changes in user base, or misuse of the models results.
• Interest in fault tolerance is rising but its overall role and frameworks for it are still developing.
• Some patterns for fault tolerance can be – and already are – used to tackle the problems caused by the ML model in the system, despite the field still developing.

				7	S	07/02/2022 10:49

Files\\Quality Assurance for AI-Based Systems~ Overview and Challenges (Introduction to Interactive Session)

No	Scopus	0.0811	6			
				1	S	04/02/2022 13:58

With the pervasive use and the dependence on AIbased systems, the quality of these systems becomes essential for their practical usage. However, quality assurance for AI-based systems is an emerging area that has not been well explored and requires collaboration between the SE and AI research communities. This paper discusses terminology and challenges on quality assurance for AI-based systems to set a baseline for that purpose. Therefore, we define basic concepts and characterize AI-based systems along the three dimensions of artifact type, process, and quality characteristics. Furthermore, we elaborate on the key challenges of (1) understandability and interpretability of AI models, (2) lack of specifications and defined requirements, (3) need for validation data and test input generation, (4) defining expected outcomes as test oracles, (5) accuracy and correctness measures, (6) non-functional properties of AI-based systems, (7) self-adaptive and self-learning characteristics, and (8) dynamic and frequently changing environments.

				2	S	04/02/2022 14:00

The knowledge and background of different communities are brought together
for developing AI-based systems. While this leads to new and innovative approaches, exciting breakthroughs, as well as a significant advancement in what can be achieved with modern AI-based systems, it also fuels the babel of terms, concepts, perceptions, and underlying assumptions and principles. For instance, the term "regression" in ML refers to regression models or regression analysis, whereas in SE it refers to regression testing. Speaking about "testing", this term is defined as the activity of executing the system to reveal defects in SE but refers to the evaluation of performance characteristics (e.g., accuracy) of a trained model with a holdout validation dataset in ML. The consequences are increasing confusion and potentially conflicting solutions for how to approach quality assurance for AI-based systems and how to tackle the associated challenges. While this paper starts from a software engineering point of view, its goal is to incorporate and discuss also many other perspectives, which eventually aggregate into a multi-dimensional big picture of quality

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On
				3	S	04/02/2022 14:01

For instance, additional quality properties of AI components and AI-based
systems have to be taken into account. Zhang et al. [5] consider the following quality properties:
– Correctness refers to the probability that an AI component gets things right. – Model relevance measures how well an AI component fits the data. – Robustness refers to the resilience of an AI component towards perturbations. – Security measures the resilience against potential harm, danger or loss made via manipulating or illegally accessing AI components.
– Data privacy refers to the ability of an AI component to preserve private data information.

				4	S	04/02/2022 14:00
				5	S	04/02/2022 14:01

Efficiency measures the construction or prediction speed of an AI component. – Fairness ensures that decisions made by AI components are in the right way and for the right reason to avoid problems in human rights, discrimination, law, and other ethical issues.
– Interpretability refers to the degree to which an observer can understand the cause of a decision made by an AI component.

				6	S	04/02/2022 14:01

In addition to outlining important concepts and terms in the previous section,
this section elaborates on the following key challenges encountered in the development of approaches for quality assurance and testing of AI-based systems.
– Understandability and interpretability of AI models – Lack of specifications and defined requirements – Need for validation data and test input generation – Defining expected outcomes as test oracles – Accuracy and correctness measures – Non-functional properties of AI-based systems – Self-adaptive and self-learning characteristics – Dynamic and frequently changing environments.

Files\\Quality Management of Machine Learning Systems

No	Scopus	0.0348	3			
				1	S	04/02/2022 13:45

In spite of an explosive growth in the raw AI technology and in consumer facing applications on the internet, its adoption in business applications has conspicuously lagged behind. For business/missioncritical systems, serious concerns about reliability and maintainability of AI applications remain. Due to the statistical nature of the output, software ‘defects’ are not well defined. Consequently, many traditional quality management techniques such as program debugging, static code analysis, functional testing, etc. have to be reevaluated. Beyond the correctness of an AI model, many other new quality attributes, such as fairness, robustness, explainability, transparency, etc. become important in delivering an AI system.

				2	S	04/02/2022 13:49

This supports the assertion that moving AI from a proof-ofconcept to real business solution is not a trivial exercise. Some common reasons cited for this result are:
– Insufficient alignment of business goals and processes to the AI technology (akin to the challenges of introducing information technology in the 1990’s).
– Lack of data strategy (i.e. “There is no AI without IA (Information Architecture)”)
– Shortage of skilled people who can combine domain knowledge and the relevant AI technology.
– Unique concerns about AI (e.g. model transparency, explainability, fairness/bias, reliability, safety, maintenance, etc.)
– Need for better engineering infrastructure for data and model provenance. As the application of AI moves to business/mission critical tasks with more
severe consequences, the need for a rigorous quality management framework becomes critical. It is bound to be very different from the practices and processes that have been in place for IT projects over many decades.

				3	S	04/02/2022 13:52

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On

Files\\Software Architecture Challenges for ML Systems

No	IEEE	0.1139	5			
				1	S	24/02/2022 14:36

operations perspectives that occur when these groups work independently [10].
III. CHALLENGES FOR ARCHITECTING ML SYSTEMS We present four categories of software architecture chal-
lenges that need to be addressed for the process depicted in Figure 2 to support ML system development, as well as their maintenance and evolution.
A. Software Architecture Practices for ML Systems Existing established software architecture practices to sup-
port design, development, and deployment of software systems [2], in addition to data-intensive system paradigms (e.g., big data analytics systems [18]), provide a foundation for architecting ML systems. A ML component can be considered a software component with characteristics that are not common in traditional software components. The behavior of a traditional software component is defined by rules programmed in code that address its QA requirements and expected response measures. However, the behavior of a ML component is defined by characteristics of the datasets it is trained with, in
addition to the system's QA concerns [24]. Therefore, software architecture practices will need to take into account how to address requirements specification, design specification, and interpretability concerns driven by datasets [9]. Software development processes, including agile software
development processes, went through an alignment stage to incorporate architecture tasks effectively. A similar adjustment will be needed to align the experimental, iterative and incremental nature that is inherent in architecting and development of ML models and ML components [1] [21]. Although continuous evolution and iterative development are not new to software architecting, the uncertainty introduced by the volatility of the data that drives ML component development is certainly not common. ML component development relies on generate-and-test approaches which make them hard to align with sprint boundaries and identification of ”done criteria” common in most software development processes. In summary, while many existing software architecture practices and design paradigms are applicable to ML systems, some will need to be adapted to account for data-dependent behavior of ML components. New practices will need to be developed to account for ML-important QAs (next section). To note is that any existing, adapted, or new software architecture practice will need to take into account perspectives of new sets of stakeholders [6] [12] [17], including data

				2	S	24/02/2022 14:36

B. Architecture Patterns and Tactics for ML-Important QAs Quality attributes (QAs) — properties used to evaluate the quality and fitness of a system to meet its business goals — drive the selection of architecture approaches, includ-
ing patterns and tactics, and consequently the structure and behavior of software systems [2]. While organization- and domain-specific business goals shape architectural and other system requirements, ML-system-specific QA concerns also play an important role in ML systems. These attributes include explainability, data centricity, verifiability, monitorability, observability, and fault tolerance, at a minimum, in addition to elevated importance of security and privacy [19]. Analysis techniques to assure their correct design and implementation will need to be developed. These attributes will also drive the development of architecture-level techniques for addressing fairness, unintended bias, and ethics, in particular to limit propagation of unintended consequences.

				3	S	24/02/2022 14:36

Understanding monitorability of ML systems requires ad-
ditional work in several areas. First, we need to understand what different monitoring techniques will be needed for data quality vs. model quality vs. software quality vs. service quality. Existing patterns and tactics for monitorability and observability will apply for some, but new ones will need to be developed as well. Second, there are opportunities to relate monitorability to self-adaptation [17]: (1) of ML:ML models self-adapt to system changes (one of the goals of MLOps), (2) for ML: ML system adapts to changes in the system that affect quality of service (QoS), and (3) by ML: system uses ML to adapt. And lastly, we need to understand

				4	S	24/02/2022 14:37

D. Co-Architecting and Co-Versioning A ML system has two software architectures that need
to be developed and sustained: (1) the architecture of the ML system as described in Figure 2 (the system that uses the ML components), and (2) the architecture of the system that supports the ML model life cycle shown in Figure 1 (the system that produces the ML model, often called the model development pipeline). This latter architecture is often neglected as models are developed in trial-and-error, experimental mode, often by people whom are not trained in software engineering [10]. We use the term co-architecting to refer to the fact that both architectures need to be developed in sync, such that design decisions are driven by both system and model requirements, as well as the perspectives of the different stakeholders and development teams. An architectural approach for the model development pipeline also promotes potential for reuse, especially for data pipeline components (i.e., components that extract and transform raw data into training data). Successful co-architecting requires additional work in three areas: (1) practices that enable synchronization and integration between the two architectures, (2) architecture representations for ML-relevant concerns (e.g., data quality, model accuracy), as well as ML-relevant components (e.g., data pipelines, model elements), and (3) architecture views for model development pipelines that reflect and communicate design decisions and concerns related to data and feature engineering (e.g., data distribution, algorithm selection, feature selection). Work in these areas needs to consider effective communication, simple representation, and visualization tools because co-architecting will likely happen in teams that combine data scientists, software engineers, and potentially other

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On
				5	S	24/02/2022 14:37

The software architecture of a system is the collection of structures that depict the behavior of the system and inform how well the system meets its business and quality goals, including how well the longevity of the system is supported from a maintenance and evolution perspective. Development, deployment, maintenance, and evolution of systems that include ML components pose different architecting challenges. In this paper, we summarized these challenges collected from researchers and practitioners through workshops, interviews, and industry engagements. Research needs to question existing software architecture concepts and practices for their fitness to support ML systems. The challenges include understanding how well existing software architecture practices support ML system development, as well as developing patterns and tactics to respond to ML-important QAs. In addition, there are architectural

Nodes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Building ML Systems and applications\MLOPS

PDF

Files\\MLOps Challenges in Multi-Organization Setup~ Experiences from Two Real-World Cases

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On
No	Scopus	0.1381	10			
				1	S	07/02/2022 11:28

While numerous proposals exist from different vendors,
perhaps the most well-known incarnation of MLOps is Continuous Delivery for Machine Learning (CD4ML) [19]. The approach formalized by ThoughtWorks for automating in an end-to-end fashion the lifecycle of machine learning applications. In CD4ML, a cross-functional team produces machine learning applications based on code, data, and models in small and safe increments that can be reproduced and reliably released at any time, in short adaptation cycles. The approach contains three distinct steps: identify and prepare the data for training, experimenting with different models to find the best performing candidate, and deploying and using the selected model in production. The work is split to an ML pipeline that works with the data, and to a deployment pipeline that deploys the result to operations (Figure 1). The above implies that there are three artifacts, in addition to
those that are required by DevOps, that need version control in MLOps: (i) different data sets used for training model and their versioning; (ii) model and its versioning; and (iii) monitoring the output of the model to detect bias and other problems.

				2	S	07/02/2022 11:29
				3	S	07/02/2022 11:30

Consequently, operations related to data seem to be the
most difficult to put into practice. In general, systems like datalakes can be used to integrate data from various sources, but if amounts of data are massive and, in addition, its owners want to protect it, this option is feasible only inside one organization.

				4	S	07/02/2022 11:31

Firstly, the model is created, and its quality assurance
activities are carried out on the hospital's premises as a shared activity between the two organizations. The mode of operation for this is based on experiments where interesting properties are identified in the dataset, which in general is often the nature of data science projects early on [1]. The rhythm for the operations is defined by these experiments. If desired, the model can be re-created with more precision in given intervals or by some other valid form of meaningful iteration. Each new iteration cycle creates a new version of the model, and it may or may not be handed over to the service provider. Secondly, the service provider is responsible for the devel-
opment and the operations of the software that are necessary to use the model as basis for collaboration between the doctors and (potential) patients.

				5	S	07/02/2022 11:31

Finally, the tool is meant to help the doctor and the patient
to discuss the risks related to a surgical operation, not to decide whether or not to perform the operation. Instead, the decision is always made by the humans, and the AI only has a supporting role in the process. Hence, the responsibility is carried by humans, not by the AI. Furthermore, in the unlikely event of the system malfunctioning and providing answers that clearly are infeasible, the doctor – an expert in such operations – is able to notice them and fix the situation.

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On
				6	S	07/02/2022 11:31

Supporting interoperability at technical, informational and governance levels, such an ecosystem is aligned with the AuroraAI vision, where it is the individuals who combine data, not the society. The use of the digital twin paradigm [8] has also been considered in this context [12], leading to citizen-level use of datasets and recommendations. Unfortunately, such an approach, relying on datasets owned by multiple organizations, does not really provide a data set that would be easily available for ML or even deeper analysis. Firstly, MyData is not automatically shared but is something that only the individuals can release in accordance to their wishes. Secondly it is not obvious which data is true and which false, as individuals themselves provide some data, and, moreover, they can manipulate some data.

				7	S	07/02/2022 11:32

Since models are concrete assets in the ML context as well as from operations perspective, they are also something that can be easily shared in AuroraAI. However, these models are only partial, as they are built by different data owners, not based on personal data that

				8	S	07/02/2022 11:33

For AuroraAI, this has meant that instead of aiming at automata that can provide recommendations for everyone, models are more targeted to individuals, who can use them to determine facts about their well-being. Moreover, based on the models and input from the user, recommendations are given to propose actions to add the observed well-being. Obviously, if an individual citizen chooses to share the results with municipalities, chances are that the individual in question will get a better, more targeted service proposals. However, sharing the results is by no means enforced, meaning that the resulting data set is heterogeneous from the society perspective.

				9	S	07/02/2022 11:33

That said, individual offices often have such systems in
place locally, as this is governed by law. Hence, they can monitor what takes place, and, at least to some extent, who accesses what. Opening such monitoring data to individuals with

				10	S	07/02/2022 11:33

A new challenge in software engineering for ML is data
related operations. These operations are related to the above to some extent, especially when data sets cannot be moved across data boundaries, but multiple organizations need to access the data. Moreover, data meshes and other forms of integrating data in pieces can complicate designing the more traditional parts of information systems, needed for such integration. In addition, when considering AuroraAI, it seems natural that different solutions might rely on different versions of data sets, for several reasons. For instance, it is possible that extensive data cleaning operations are needed for some applications, meaning that executing such operations frequently is impossible. Similarly, it might be so that the data must be from the same temporal range, and otherwise the operations make no sense. Similar complications are reflected to training ML models based on such data sets, as well as to monitoring how well the models work once they have been deployed. For operationalizing all the above in practice, the same skill
gap as for starting to use MLOps within a single organization is valid – indeed the same actions need to be taken. However, this time some of the issues are more difficult to reconcile, because the organizations may have different modes of operation and different organization cultures, as demonstrated in the Oravizio case. Moreover, restrictions, such as those related to privacy or certification, may exist on either side of the boundary, which adds an additional layer of complexity to the design. This has also been identified as a direction for future work, especially from the perspective of governance, auditing, and regulations [22]. In general, to successfully perform multi-organization MLOps, we need patters of integration that help us in the process. Inspiration for these can be found from system integration [10] as well as legality patterns, proposed for open source [9]. In fact, both solutions we have used in the examples of the paper are analogous to patterns of [9] – Oravizio uses the ML model as an Evaluator, and in AuroraAI, User delegation helps to combine data that can only be accessed by the user as a whole. The definition of such patterns remains future work, with some ideas already proposed in [17]. Finally, based on both case studies reported in this paper, it seems that if there is the will, there often is a way

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On

Files\\Towards MLOps~ A Framework and Maturity Model

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On
No	Scopus	0.0126	1			
				1	S	03/02/2022 10:17

Challenges associated with MLOps In our own previous research [16] [17], we have identified
a number of challenges when it comes to the business case, data, modeling and deployment of ML or Deep Learning (DL) models. These include high AI costs and expectations, fewer data scientists, need for large datasets, privacy concerns and noisy data, lack of domain experts, labeling issues, increasing feature complexity, improper feature selection, introduction of bias when experimenting with models, highly complex DL models, need for deep DL knowledge, difficulty in determining final model, model execution environment, more hyperparameter settings, and verification and validation. It also includes less DL deployment, integration issues, internal deployment, need for an understandable model, training-serving skew, enduser communication, model drifts, and maintaining robustness. Some of the challenges in MLOps practice [5] include tracking and comparing experiments, lack of version control, difficulty in deploying models, insufficient purchasing budgets and a challenging regulatory environment.

Nodes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Data Engineering\Data cleaning

PDF

Files\\Data Cleaning for Accurate, Fair, and Robust Models~ A Big Data - AI Integration Approach

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On
No	Scopus	0.1077	6			
				1	S	07/02/2022 23:20

. While many techniques have been proposed to improve the model training process (in-processing approach) or the trained model itself (post-processing), we argue that the most effective method is to clean the root cause of error: the data the model is trained on (pre-processing). Historically, there are at least three research communities that have been separately studying this problem: data management, machine learning (model fairness), and security. Although a significant amount of research has been done by each community, ultimately the same datasets must be preprocessed, and there is little understanding how the techniques relate to each other

				2	S	07/02/2022 23:20

We contend that it is time to extend the notion of data cleaning for modern machine learning needs. We identify dependencies among the data preprocessing techniques and propose MLClean, a unified data cleaning framework that integrates the techniques and helps train accurate and fair models. This work is part of a broader trend of Big data – Artificial Intelligence

				3	S	07/02/2022 23:21

We compare and identify dependencies among the three data preprocessing techniques and discuss how data cleaning can possibly be extended to the other preprocessing techniques.
2.1 Traditional Data Cleaning Data cleaning [4] originates from the data management community and has been studied for decades. Traditionally, there is a focus on cleaning structured data with schema at scale where integrity constraints, denial constraints, and functional dependencies need to be satisfied. In addition, duplicates must be removed, and values need to be corrected to be within certain ranges or to exist in external data sources. More recently, there are efforts to improve machine learning accuracy [5] and data validation techniques for machine learning pipelines [10]. However, these techniques do not resolve the pressing issues of model fairness or model robustness against adversarial data.

				4	S	07/02/2022 23:21

Data Sanitization The machine learning and security communities are actively
studying the problem of robust machine learning against adversarial data in critical applications including spam filtering, autonomous driving, and cybersecurity. A major problem is that the training data is often collected from external data sources, which are vulnerable to attacks by malicious actors [9]. A popular solution is to make the model training more robust. Another approach that is gaining interest is sanitizing the poisoned data before it is used in training. Data poisoning attacks have recently become more sophisticated [9], and there is an arms race on developing better defenses to stop them as well. Data poisoning can also be done on the test data where the same sanitization techniques can apply. Data sanitization may conflict with data cleaning.

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On
				5	S	07/02/2022 23:22

MLCLEAN
Since data cleaning, unfairness mitigation, and data sanitization are ultimately preprocessing the same dataset, it makes sense to unify them. The naïve approach of applying each technique independently in any sequence can be problematic for several reasons. Simply ignoring the dependencies between preprocessing techniques may result in incorrect results. For example, ifwe reweigh examples and then attempt to remove duplicates, then the reweighing may need to be done again to ensure fairness. Moreover, running one operation at a time may have efficiency issues due to redundant operations on the data.
3.1 Basic Architecture
We present MLClean, an extended data cleaning framework that takes into account the dependencies of the three preprocessing techniques and integrates them to produce clean, unbiased, and sanitized data (see architecture in Figure 1). Data sanitization can be viewed as an extreme version ofdata cleaning and thus be executed together in one component. The unfairness mitigation component comes afterwards because, while data sanitization and cleaning may affect the bias of data, reweighing examples only changes the example weights and does not effect the correctness of sanitization and cleaning on the other features.

				6	S	07/02/2022 23:22

Files\\Data collection and quality challenges for deep learning

No	Web of science	0.0596	4			
				1	S	07/02/2022 23:17

Compared to traditional machine learning, there is less need for feature engineering, but more need for significant amounts of data. We thus go through stateof-the-art data collection techniques for machine learning. Then, we cover data validation and cleaning techniques for improving data quality. Even if the data is still problematic, hope is not lost, and we cover fair and robust training techniques for handling data bias and errors. We believe that the data management community is well poised to lead the research in these directions. The presenters have extensive experience in developing machine learning platforms and publishing papers in top-tier database, data mining, and machine learning venues.

				2	S	14/02/2022 14:42

While there is a vast literature on data cleaning, not all of the techniques are beneficial to machine learning [8]. In addition, recent machine learning issues including data poisoning need to be addressed as well. Even after carefully preparing the data, the data quality may still be problematic, and we need to cope with biased, dirty, or missing data using fair and robust model training [14, 15].

				3	S	07/02/2022 23:18

Data cleaning has a long history of removing various
well-defined errors by satisfying integrity constraints including key constraints, domain constraints, referential integrity constraints, and functional dependencies. Unfortunately, only focusing on fixing the data does not necessarily guarantee the best model accuracy. We cover the recent CleanML [8] work, which systematically studies the impact of data cleaning on the accuracy of the model trained on that data. The conclusions are twofold: data cleaning does not necessarily improve the model accuracy, and performing model selection can at least reduce any negative effects where the data cleaning may harm model accuracy. Hence, we cover recent data cleaning techniques that are specifically geared towards improving model accuracy.

				4	S	14/02/2022 14:07

Data poisoning has recently become a serious issue be-
cause changing a fraction of the training data, which may come from an untrusted source, may alter the model's behavior. Compared to dirty data, there is a malicious intention of making the model fail. Early work focused on specific applications like spam detection and sensors. More recent studies are more general, but still tend to focus on specific models. It is unclear if there will be anything close to a unifying solution. The notion of data sanitization was introduced in 2008 [4] where attacks were assumed to occur in relatively confined time intervals, and the sanitization techniques used training metadata. More recently, adversarial machine learning, which attempts to fool models through malicious inputs (e.g., adversarial images), has become one of the most popular topics in machine learning.

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On

Nodes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Data Engineering\Data managment

PDF

Files\\Juneau~ data lake management for Jupyter

No	Web of science	0.0419	2			
				1	S	07/02/2022 15:31

A challenge lies in transitioning from exploratory data analy-
sis, possibly over a small amount of data, to something that can be regularized into a production workflow with full reproducibility and much larger datasets. Towards this goal, recent work has proposed using notebooks as a way of encoding repeated computational workflows [9], and others have developed extensions to ensure the code within notebooks is fully versioned and reproducible [10, 1, 6]. However, we argue that the next step must be to look not at note-
books as documents of code steps that access and produce data files — but rather as compilations of (possibly shared, possibly param-eterized) computational steps operating on objects in a data lake. We seek to accelerate and regularize data science tasks by finding and recommending data related to current objects of interest to the user. We do this by tracking the relationships between data sets, data products, and code [5]. With the appropriate indexing and search capabilities, data import and data cleaning steps are made visible to future users to be reused; data scientists may find other related datasets with similar history provenance; users are able to query, based on a given source table or intermediate result, whether someone else has already linked two datasets or extracted sets of features. Ultimately, just as shared libraries and open-source repositories have accelerated and improved software engineering — reusable datasets, schemas, and computational workflow steps may improve the quality of data engineering.

				2	S	07/02/2022 15:33

Files\\Shuffler~ A Large Scale Data Management Tool for Machine Learning in Computer Vision

No	Scopus	0.0455	5			
				1	S	04/02/2022 13:05

Datasets in the computer vision academic research community are primarily static. Once a dataset is accepted as a benchmark for a computer vision task, researchers working on this task will not alter it in order to make their results reproducible. At the same time, when exploring new tasks and new applications, datasets tend to be an ever changing entity. A practitioner may combine existing public datasets, filter images or objects in them, change annotations or add new ones to fit a task at hand, visualize sample images, or perhaps output statistics in the form of text or plots.

				2	S	04/02/2022 13:06

Given that ML and deep learning call for large volumes of data to produce satisfactory results, it is no surprise that the resulting data and software management associated to dealing with live datasets can be quite complex. As far as we know, there is no flexible, publicly available instrument to facilitate manipulating image data and their annotations throughout a ML pipeline.

				3	S	04/02/2022 13:09

In the computer vision academic community, day-to-day work emphasizes primarily algorithms rather than data. From this point of view, annotated image datasets are ideally built once and remain fixed. This approach allows the community to use datasets as benchmarks. Researchers choose to store these datasets in formats that are most common and fast to load for machine learning (ML) packages. In contrast, for a data scientist in industry, the task is not necessarily to improve an algorithm, but rather to try different algorithms and tasks on various partitions and modifications of the same dataset. In this case, a dataset is not considered static, but rather constantly altered to fit the task at hand. In turn, multiple versions of the same dataset need to co-exist in a centralized or a distributed storage system. Ideally, a practitioner would want 1) a simple way to manipulate image data and its annotations, and 2) a file format that allows to store multiple copies of the annotation set in an organized and efficient way and to inspect them manually. Data manipulation tools are sometimes packaged with a dataset,
but they typically allow to perform only a limited number of operations only on that particular dataset and often for a single programming language

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On
				4	S	04/02/2022 13:10

Datasets typically are in a custom format, which usually includes
an image and an annotation file in one of the following formats: xml, txt, or json. Table 1 presents an overview of several popular object detection datasets in the area of computer vision and the formats of the associated annotation files. On the one hand, these formats are human-readable, but on the other hand, quite slow to load. Additionally, changing annotations and saving them as a copy means duplicating the whole directory with the annotation files, which is inconvenient and slow. Many development kits cache annotations by serializing them with formats such as pickle1 or protobuf2. Such formats are easy to load by a machine

				5	S	04/02/2022 13:11

To sum up, we consider (Figure 2) a typical data preparation
workflow of a computer vision practitioner to consist of three steps: 1) download or collect a dataset, 2) modify annotations, and 3) serialize the dataset. We further consider a common situation when multiple modifications of annotations are used. Modifications could be a chain of trivial tasks, for example, removing objects at image boundaries and then increasing the size ofbounding boxes. We note 1) the lack ofsoftware for manipulating image data and annotations, and 2) a convenient format to store annotations.

Nodes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Data Engineering\Data preprocessing

PDF

Files\\A hybrid method for missing value imputation

No	ACM Digital library	0.0627	4			
				1	S	11/02/2022 13:00

Missing values are a common incurrence in a great number of real-world datasets, emerging from diverse domains of interest. In research, missing data constitute a significant problem as it can affect the conclusions drawn from them. Considering this, the difficulty of data preprocessing is increasing as selecting an inappropriate way to handle missing information can lead to untrustworthy results. Unfortunately, like in most cases in Machine Learning, there is not a single solution that fits in every task related to the problem. For this reason, many strategies have been proposed to successfully deal with this issue. One of the most well-known, besides efficient, is imputation. Replacing a missing value with an estimation apparently eliminates the problem and provides complete datasets but the difficulty shifts in selecting the right method to impute missing values.

				2	S	11/02/2022 13:00

A familiar problem to Machine Learning researchers and data
analysts is the occurrence of missing data, which reside in almost every dataset, setting obstacles in the stage of data preprocessing. Missing data, also referred as missing values, occur when no value is stored for the variable of an attribute, leaving the actual value of the observation unknown. The most common scenario is that multiple values, usually in different attributes are missing in a single dataset, leading to the absence of a not negligible subset of it. The fact that a portion of the actual dataset is missing means that the amount of information that can be drawn from the data is reduced, a mater that strongly affects the ability to understand and explain the phenomenon of interest and raises concern about the reliability of the study results [1].

				3	S	11/02/2022 13:02

Legitimate missing values, due to their nature are easier to deal with and in most times there is no need to employ sophisticated methods, like imputation, to deal with them. Moreover, in some cases the missing values belong to this category can provide the researchers with useful information about the reliability of the questionnaire. Unfortunately, not all missing data belong to the previous category. Illegitimately missing data can be found in all kinds of datasets and can be caused by numerous factors.

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On
				4	S	11/02/2022 13:02

The missing data mechanism affects how missing values bias
the results of a study, so it is essential to know its type in order to choose the most appropriate approach to deal with them. Missing data can be categorized into three major categories depending on the mechanism causing them [4]: •
2 Related work The stage of data preprocessing is fundamental in Machine
Learning tasks as can influence remarkably the quality of the extracted results. Considering this matter of fact, it is clear why dealing with missing values is a very active research field. Although the related literature is rich and plenty of work had been done on this specific issue, unfortunately, there is not a single way that can handle every individual case that lie in this field. Missing values reside in datasets emerging from different domains, and as long as each of them has its specific features it is obvious that the nature of missing data that exist in a dataset has a principal role in the selection of the right treatment approach. The methods that already have been proposed to deal with missing values can be clustered in two categories. The first and simplest method, suggests to ignore or discard missing data. The second one suggests to replace the missing value, with a new one, or in other words to impute it. Both approaches are discussed below.
Considering the first approach, ignoring the missing data in
Missing completely at random (MCAR): The missing values occur completely at random and are distributed evenly among the observations. In other words, all the observations share an equal probability to be missed. The reason of missingness is not related to the observed variables or unobservable parameters of interest. In this case, missing values are a random subset of the dataset, and no other data (missing or observed) are related to them.
• Missing at random (MAR): The MAR values are not related to the missing data, but are related to some of the observed data. This means that missing values are related to one or more variables of the dataset. To be more specific, a value is MAR, when the probability to be missing depends only on available information. MAR values are more common than MCAR.
•
Missing not at random (MNAR): The value of missing data is related to the reason of missingness. The phenomenon that all the values of an attribute are missing due to their values is referred to as censoring [13], but in real-world scenarios is extremely hard to take place.

Files\\An Empirical Study of the Impact of Data Splitting Decisions on the Performance of AIOps Solutions

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On
No	Google Scholar	0.0295	6			
				1	S	10/02/2022 10:58

Despite the breakthroughs in ML models and their applications in AIOps, there are still challenges preventing the integration of such ML models into software systems [41], such as the challenges in model evaluation and model evolution [3, 73]. One of the main reasons is that ML experts usually focus on tuning the MLmodel performance instead ofmaintaining model behavior after deploying in the field [41]. Hence, software engineering for machine learning has become an emerging topic that aims to manage the lifecycle ofmachine learning models (i.e., training, testing, deploying, evolving, etc.) [3, 41, 63]. Within the lifecycle of ML models, making appropriate decisions for data splitting (e.g., splitting data into training and validation sets) is particularly challenging, even for ML experts [38, 63]. For example, ML experts highlight the importance of data splitting in ML modeling [63] and advocate the introduction of engineering processes for data splitting [38]. In particular, in the context ofAIOps, ML modeling faces three data splitting (DS)-related challenges during the process

				2	S	10/02/2022 10:59

• Imbalanced data: Operation data is often very imbalanced [5, 24, 49, 51, 57], which challenges AIOps modeling, as the models tend to make a more accurate prediction on the majority class while performing poorly on the minority class [44, 80, 89]. Such a challenge requires the application of data rebalancing techniques (e.g., over-sampling, under-sampling, SMOTE [13], ROSE [54]) to make the modeled classes more balanced (i.e., splitting the data ofdifferent classes to achieve a better balance between the classes) [44, 80] or using cost-sensitive models [1, 15, 35].
• Data leakage: Prior studies (e.g., References [5, 24, 57]) in AIOps randomly split operation data into training and validation data. However, such a splitting strategy may risk data leakage, i.e., leak information in the validation data that should not be available for model training into the training data, which may introduce bias and result inmisleading evaluation results [39, 40, 65, 72]. For example, in a recent Kaggle competition [74], the leakage of the future information into the training features cause the model to make unrealistic good predictions that could not reflect the actual model performance in a practical setting.
• Concept drift: Over time, the distribution of the operation data and the relationship between the variables in the data may be constantly evolving [17, 49, 51] (a.k.a. concept drift [64, 84–86]). Concept drift may lead to obsolescence of the models trained on historical data, i.e., a model trained on outdated data may perform poorly on new data.

				3	S	10/02/2022 10:59

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On
				4	S	10/02/2022 11:01

The Challenge of Imbalanced Data: Imbalanced data is a common problem in the ma-
chine learning community. It arises when one of the classes is severely underrepresented in the dataset. [32]. Imbalanced data could cause models to focus on the majority class and ignore the rare events, which heavily compromises the process of learning [44]. The machine learning community usually addresses the issue in two ways. One way is to apply
data rebalancing techniques, most simply, oversampling the minority class or under-sampling the majority class. There are also approaches that blend the two sampling strategy like SMOTE (Synthetic Minority Over-sampling TEchnique) [13], and approaches that combine oversampling with the generation of artificial data like ROSE (Random OverSampling Examples) [59]. As a result, the modeled classes are more balanced and may produce more predictive models. Other than resampling techniques that balance the sample classes, researchers also design ML
models specially optimized for the imbalanced data issue by assigning distinct costs to the training samples. For example, Arya et al. [35] propose a cost-sensitive support vector machine algorithm that provides superior generalization performance compared to conventional SVM on imbalanced data; deep learning approaches can also tackle the imbalanced data problem with a weighted backpropagation [15] or a weighed form of categorical cross-entropy [1]. Besides, updatable classification algorithms can also be a viable approach in handling imbalanced data. Ming et al. [78]report that updatable classification algorithms, which update the training set incrementally to take

				5	S	10/02/2022 11:01

The Challenge of Data Leakage: Data leakage is the introduction of information in the
training data that should not be available for model training and can lead to the bias of model evaluation [39, 40, 65, 72]. The creation of such unexpected additional information in the training data would enable the models to use the future data to predict the past data [49, 51, 89], and therefore cause it to make unrealistically good predictions that could not reflect the practical performance. Leakage is a pervasive challenge in applied machine learning, causing models to over-represent their generalization error and often rendering them useless in the real world. For example, leakage of the future information into the training features are reported in many Kaggle competitions, including a recent one in a prostate cancer dataset [74]. Prior works [39, 72] suggest that when there are risks of such data leakage, time-based splitting of training and validating data splitting (i.e., splitting the data based on their time sequence) should be used over a random-based splitting strategy. In this work, we study the existence of data leakage in the context of AIOps, which has not been explored before. We also evaluate the impact of different splitting strategies (e.g., time-based splitting) on data leakage.

				6	S	10/02/2022 11:01

The Challenge of Concept Drift: In machine learning and data mining, the distribution of
the data and the relationship between the variables may evolve over time, which is known as concept drift [64, 84–86]. Concept drift may lead to obsolescence of models trained on previous data and negatively impact the performance. To mitigate the impact of concept drift, prior works propose approaches for detecting concept drift [26, 30, 64, 87] and handling concept drift [9, 12, 21, 28, 32, 60, 61, 77, 85]. For example, Nishida et al. [64] propose a concept drift detection method using statistical testing. It assumes that the prediction accuracy on the data from a recent time window would be equal to the overall accuracy if the target concept is stationary, and a significant decrease in the recent accuracy suggests a concept drift. When there is concept drift, aside from retraining a model from scratch, online learning updates the current model using the most recent data incrementally. Such model process input examples one-by-one and update

Files\\High Performance Data Engineering Everywhere

No	IEEE	0.0155	2			
				1	S	07/02/2022 16:38

The amazing advances being made in the fields of
machine and deep learning are a highlight of the Big Data era for both enterprise and research communities. Modern applications require resources beyond a single node's ability to provide. However this is just a small part of the issues facing the overall data processing environment, which must also support a raft of data engineering for pre- and post-data processing, communication, and system integration. An important requirement of data analytics tools is to be able to easily integrate with existing frameworks in a multitude of languages, thereby increasing user productivity and efficiency. All this demands an efficient and highly distributed integrated approach for data processing, yet many of today's popular data analytics tools are

				2	S	07/02/2022 16:39

Large-scale data processing/engineering has gone through remarkable transformations over the past few decades. Developing fast and efficient Extract, Transform and Load frameworks on commodity cloud hardware has taken center stage in handling the information explosion and Big Data. Subsequently, we have seen a wide adoption of Big Data frameworks from Apache Hadoop [1], Twister2 [2], and Apache Spark [3] to Apache Flink [4] in both enterprise and research communities. Today, Artificial Intelligence (AI) and Machine Learning (ML) have further broadened the scope of data engineering,

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On

Files\\Inspector gadget~ a data programming-based labeling system for industrial images

No	Web of science	0.0249	3			
				1	S	07/02/2022 15:43

As machine learning for images becomes democratized in the Software 2.0 era, one of the serious bottlenecks is securing enough labeled data for training. This problem is especially critical in a manufacturing setting where smart factories rely on machine learning for product quality control by analyzing industrial images. Such images are typically large and may only need to be partially analyzed where only a small portion is problematic (e.g., identifying defects on a surface). Since manual labeling these images is expensive, weak supervision is an attractive alternative where the idea is to generate weak labels that are not perfect, but can be produced at scale. Data programming is a recent paradigm in this category where it uses human knowledge in the form of labeling functions and combines them into a generative model. Data programming has been successful in applications based on text or structured data and can also be applied to images usually if one can find a way to convert them into structured data. In

				2	S	07/02/2022 15:44

We focus on the problem of scalable labeling for classification
where large images are partially analyzed, and there are few or no labels to start with. Although many companies face this problem, it has not been studied enough. Based on a collaboration with a large manufacturing company, we provide the following running example. Suppose there is a smart factory application where product images are analyzed for quality control (Figure 1). These images taken from industrial cameras usually have high-resolution. The goal is to look at each image and tell if there are certain defects (e.g., identify scratches, bubbles, and stampings). For convenience, we hereafter use the term defect to mean a part of an image of interest.

				3	S	07/02/2022 15:44

Among the possible methods for data labeling (see an extensive
survey [32]), weak supervision is an important branch of research where the idea is to semi-automatically generate labels that are not perfect like manual ones. Thus, these generated labels are called weak labels, but they have reasonable quality where the quantity compensates for the quality. Data programming [30] is a representative weak supervision technique ofemploying humans to develop labeling functions (LFs) that individually perform labeling (e.g., identify a person riding a bike), perhaps not accurately. However, the combination of inaccurate LFs into a generative model results in probabilistic labels with reasonable quality. These weak labels can then be used to train an end discriminative model.

Files\\Software engineering for artificial intelligence and machine learning software~ A systematic literature review

No	Google Scholar	0.0086	2			
				1	S	23/02/2022 19:42

Fig. 10 presents a hierarchy graph that shows the number of references from different
codes with the categories most referenced in the data. The more coding a category has, the larger its area. In addition, the subcategories (the child codes) are grouped into the parent category. Below, we present the categories of challenges ordered by their popularity, including Software Testing (30 references) and AI Software Quality (27). Followed by the categories of Model Development (16), Data Management (16), Project Management (15), Infrastructure (14), and Requirements Engineering (13). The categories of 10 to 6 references were AI Engineering (10), Architecture Design (8), and Model Implementation (6). The categories that had up to two references were Integration (2), Operational Support (1), and Education (1). Table B.18 presents the challenges faced by professionals in the development of AI/ML systems, and in it we present the most evident categories and subcategories, with the highest number of citations

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On
				2	S	03/02/2022 15:57

State-of-the-art AI/ML systems rely on high-effort data management tasks, such as data
exploration, data preparation, and data cleaning. Challenges regarding the data collection, processing, data availability, and quality are highlighted in our primary studies. The lack of data, the lack of values, the delay in sending data, the lack of metadata, the granularity of data, the scarcity of different samples are challenges related to the availability of data for ML projects. Other challenges are data manipulation and deviation, preparing the data set that includes data dependency, data quality, and data integration with various sources. In addition, the modelling of this data is one of the challenges related to data pre-processing, regarding data cleanliness, categorical data/sequence. In real-life applications, the following are common data problems: lack of metadata missing values data granularity integration data from multiple sources shortage of diverse samples design and management of the database, data lake quality of training data vs. real data
One study has highlighted the importance of data dependency, and states that data
dependencies cost more than code dependencies in AI/ML systems, i.e. unstable or underutilized data dependencies (Sculley et al., 2015). Another issue mentioned is data drift, meaning that the statistical properties of predicting variables changing in an unforeseen way (Lwakatare et al., 2019; Munappy et al., 2019). Handling of data drifts in uploaded data, invalidation of models, e.g., due to changes in data sources, and the need to monitor models in production for staleness are problems mentioned in Lwakatare et al. (2019).
Take away 5: AI development processes need to integrate infrastructures, processes and tools for managing data as their integral parts. It

Files\\Why is Developing Machine Learning Applications Challenging~ A Study on Stack Overflow Posts

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On
No	Web of science	0.0047	3			
				1	S	31/01/2022 14:44

(3) the data preprocessing and model deployment phases are where most of the challenges lay; and (4) addressing most of these challenges require more ML implementation knowledge than ML conceptual knowledge.

				2	S	23/02/2022 20:02

Data Pre-processing and Manipulation (DP)
We assume the developer is preparing his data for a ML model(s). Questions about data loading, data accessing, data cleaning, data splitting, data format changing, data labelling, data imbalance issues, data normalization, etc.

				3	S	23/02/2022 20:03

The most challenging ML topics show difficulty with data and feature preprocessing, environment setup, and model deployment.

Nodes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Data Engineering\Data validation

PDF

Files\\Continuous validation for data analytics systems

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On
No	Scopus	0.0457	3			
				1	S	07/02/2022 23:27

This trend continues through the lifecycle, into what we call 'devUsage': continuous usage validation. In addition to ensuring systems meet user needs, organisations continuously validate their legal and ethical use. The rise of end-user programming and multi-sided platforms exacerbate validation challenges. A separate trend is the specialisation of software engineering for technical domains, including data analytics. This domain has specific validation challenges. We must validate the accuracy of statistical models, but also whether they have illegal or unethical biases. Usage needs addressed by machine learning are sometimes not specifiable in the traditional sense, and statistical models are often 'black boxes'. We describe future research to investigate solutions to these devUsage challenges for data

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On
				2	S	23/02/2022 20:25

SE is increasingly specialised [12]. A clear example of this
is in the development of data analytics systems1 ('SE4ML'). Statistical machine learning (hence 'ML') lead in the integration with development practices for data analytics systems, but is now often combined with techniques from operations research and AI. As ML moved from research to widespread industrial application, there was a realisation that the bespoke algorithms written for academic publication were not necessarily scalable for large data sets nor maintainable for evolving data schemas and analysis purposes. Moreover, in industrial application there are new development artefacts to be managed, including learned statistical models, and training data sets. Since 2015, SE4ML has adapted conventional SE practices and technologies, and created new ones

				3	S	23/02/2022 20:26

Data analytics systems also have a new validation goal:
model accuracy, also called statistical validity. Does a model created by ML really reflect the situation in the world? When reusing data, is sample population and data collection instruments that were used still appropriate? Accuracy is fundamental to validating user needs, but is also critical for ethical assessment and legal probity. Validating model accuracy can be complicated by difficulties with interpretation. In statistics, Simpson's paradox [14] is a well-known example where associations between variables can be reversed under different groupings. These threats can defeat validation.

Files\\Data collection and quality challenges for deep learning

No	Web of science	0.0136	1			
				1	S	07/02/2022 23:18

While there is a plethora of data visualization techniques,
we focus on the ones that are most relevant to machine learning. Facets, a component of TFX, shows various statistics of datasets that are relevant for machine learning. More advanced tools include SeeDB [17], which can repeatedly generate possible visualizations that are of interest. This approach has the problem of false positives, so hypothesis testing started to be used in systems like CUDE [19] to guarantee the statistical significance of the findings. Data validation focuses on finding problems in the data that affect the machine learning pipeline. TensorFlow Data

Files\\On the experiences of adopting automated data validation in an industrial machine learning project

No	ACM Digital library	0.0550	7			
				1	S	04/02/2022 22:31

Data errors are a common challenge in
machine learning (ML) projects and generally cause significant performance degradation in ML-enabled software systems. To ensure early detection of erroneous data and avoid training ML models using bad data, research and industrial practice suggest incorporating a data validation process and tool in ML system development process. Aim: The study investigates the adoption of a data validation
process and tool in industrial ML projects. The data validation process demands significant engineering resources for tool development and maintenance. Thus, it is important to identify the best practices for their adoption especially by development teams that are in the early phases of deploying ML-enabled software systems.

				2	S	04/02/2022 22:34

Data errors are common and can be difficult to detect when
developing and operating ML-enabled software systems [2, 3, 4]. For companies, data errors can result in significant losses in business value. For example, LinkedIn observed financial losses and had to put huge efforts to detect data errors in their job recommendations platform [5]. Poor visibility of complex data dependencies, errors in application code, drifts in sensor data, gaps in data due to network connection problems are among the causes of data errors [6, 7, 8]. Understanding the different types of data errors and their effects on ML projects is important because literature shows that unnecessary data cleaning can be wasteful and harmful to the training of ML models [9]. To handle data errors in ML projects, research and industrial
practice suggest integration of data validation tools into the development process of ML-enabled systems1 instead of only relying on data scientists to manually check the quality of the data [10, 2, 3, 11, 12]. Important data quality dimensions of consideration are with respect to accuracy, completeness, consistency, timeliness [3, 13]. The data validation tools are particularly useful when dealing continuously with large scale data [2, 11, 3, 5]. The data validation process is also considered an approach to testing ML-enabled software systems [14].

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On
				3	S	04/02/2022 22:35

Data validation process in ML projects In most commercial ML systems, deployed ML models
are continuously retrained in order to adapt to environmental changes [2, 15]. When retraining the ML models, new training data collected at inference time can have different distribution due to various reasons, like bugs in application code [2, 16]. Differences in data distribution at training and inference, also called training-serving skew, is one form of data errors in ML projects. When the erroneous data is not detected, ML models are retrained on problematic data and can result to performance degradation of an ML system [2, 16]. Furthermore, it is rare that training datasets collected from many different sources at different time periods would always have the same exact structure and distribution [16]. Data validation in ML projects is the process of ensuring
the high quality of data that is fed into the ML algorithm(s). The aim is to continuously check and monitor the data in order to assess its quality and identify underlying issues in data quality [2, 3]. Recently, studies have demonstrated that

				4	S	04/02/2022 22:35

Data validation tools in ML projects Ehrlinger et. al. [13] conducted a state-of-the-art survey of
data quality systems (both commercial and open-source), and investigated their measurement and monitoring functionalities in order to determine how data quality is measured and monitored. While their survey did not include other tools identified by this study (discussed in next paragraphs), several limitations are reported, including implementation errors and narrow coverage of data quality metrics for important quality dimensions [13]. In addition, their study did not report the actual use of the data quality tools in industrial ML projects [13]. We identify and present studies that discuss the use of data validation tools in industrial ML projects.

				5	S	04/02/2022 22:36

A tool called Data Linter (adopting the concept of code
lint in SE) is used to automatically inspect training data and suggest ways in which features can be transformed into suitable data representation [11]. The assumption is that data can be valid but not in a representation that the ML model can best learn from, e.g. a timestamp encoded as a string. Three types of data lints that can be detected by the tool are miscoding lints (e.g. number as string), lints for outliers and scaling (e.g. tailed distribution detectors) and packaging error lints (e.g. duplicate values). Technically, the data linter tool inspects training dataset's summary statistics, examines individual examples and names given to the data features. One main limitation of the data linter tool is that it does not allow users to configure and select a set of specific lint detectors to run. As a result, the latter affect tool performance especially for large and medium scale dataset [11].The tool does not provide support for data transformation, rather the user has to manually perform the transformations. This is in addition to the lack of proper documentation and discontinued support

				6	S	04/02/2022 22:37

Overall, studies do not provide experiences of adopting
a data validation process and tool by development different teams. The tools presented are also developed by dedicated teams in large companies with several years of experience in deploying to production several ML projects. The few studies that share experiences show slow and poor early adoption with several development iterations [5]. For companies that are in the early stages of deploying ML components to production and from the embedded domain, learning from these experiences is important to help systematize the adoption with minimum resources. This is because the data validation process and tools consume huge amounts of engineering resources

				7	S	04/02/2022 22:39

C. Barriers (RQ3) The barriers of adopting data validation include 1) limited
flexibility ofdata validation tool e.g. in terms ofease ofadding new tests, and 2) limited support for the existing technology stack of ML system development process while also ensuring low learning curve.

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On

Nodes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Data Engineering\Data versioning

PDF

Files\\On the Co-evolution of ML Pipelines and Source Code - Empirical Study of DVC Projects

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On
No	Web of science	0.0311	6			
				1	S	04/02/2022 22:57

The growing popularity of machine learn-
ing (ML) applications has led to the introduction of software engineering tools such as Data Versioning Control (DVC), MLFlow and Pachyderm that enable versioning ML data, models, pipelines and model evaluation metrics. Since these versioned ML artifacts need to be synchronized not only with each other, but also with the source and test code of the software applications into which the models are integrated, prior findings on co-evolution and coupling between software artifacts might need to be revisited.

				2	S	04/02/2022 23:01

this new generation of tools, this paper aims to empirically study the prevalence of ML pipelines in open source projects, as well as the amount of maintenance effort involved. Previous studies on non-ML projects have shown that frequent changes to source code might require corresponding changes to other software artifacts such as build files [10] and infrastructure-as-code files (IaC) [11] (or vice versa), causing overhead to developers. In the case of ML projects, changes to data and/or model pipelines might induce similar overhead due to the conceptual coupling between data, model and release pipelines

				3	S	04/02/2022 23:03

Association Rules. To measure the coupling between
DVC files and other project files, we use association rules, similar to earlier papers in this field [10], [11]. Such an association rule is of the form A⇒B, describing the possible coupling of changes to file type A (e.g., “source code”) implying changes to file type B (e.g., “DVC data file”). We use the conventional [21] metrics of “Support” (Supp), “Confidence” (Conf) and “Interest” (Lift) to measure the importance of an association rule. Supp(A) indicates the frequency of appearance of A, while Conf(A⇒B) indicates the percentage of times a change of A will happen together (“is coupled”) with a change of B.

				4	S	04/02/2022 23:02

				5	S	04/02/2022 23:03

Pipeline Complexity Analysis In order to estimate the overhead that pipeline descrip-
tions represent for data engineers/scientists and developers, we use two measures of pipeline complexity, i.e., McCabe (graph structure complexity of pipelines) and Halstead (effort to understand the textual form of the .dvc pipeline specification files).

				6	S	04/02/2022 23:05

Coupling between DVC and software artifacts are much stronger than would be expected by chance, with one out of four PRs changing source code, and one out of two PRs changing tests, requiring changes to pipeline files.

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On

Nodes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\Data Engineering\Dataset creation

PDF

Files\\Achiever or explorer~ gamifying the creation process of training data for machine learning

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On
No	ACM Digital library	0.0224	5			
				1	S	10/02/2022 12:06

The creation of necessary labels is usually performed with the aid of humans. Due to the necessary amount of training data the creation process is typically highly repetitive and quickly turns into a rather unexciting, demotivating task for the annotator.

				2	S	10/02/2022 12:07

Dangers of Gami cation As gami cation makes use of game elements, it is necessary to keep in mind that with these elements some of their risks might also be adopted. One way to approach this topic has been executed

				3	S	10/02/2022 12:07

by Callan et al. [6], where ten ctive scenarios of gami cation are presented which have been wrongly established in businesses. Recurring problems were a lack of goal-orientation, unsuitable game elements and rewarding, and the danger of revealing too much information to the employees which they might attempt to use for their bene t. Furthermore, the term addiction is mentioned in this context

				4	S	10/02/2022 12:08

The company's existing annotation tool is a multi-user web ap-
plication prototype which o ers registered users a sophisticated annotation environment for collections ofimages (typically scanned documents). The annotation tasks are of four di erent types: • handwriting annotation, where annotators are given an image of a handwritten sequence of letters and numbers which they have to type,
• document classi cation, where annotators need to classify parts of a document, e. g., to mark tables inside a form using semantic bounding boxes,
• classi cation, where annotators are asked to identify a given object, e. g., if an image contains a number,
• natural language processing (NLP), where annotators are asked to assign semantic meaning to words, for example, to mark all persons in a given text.

				5	S	10/02/2022 12:08

• "I nd labeling tasks tiresome" (65% agreed, M=0.7, SD=1.117) • "Iwould like to be able to see howwell Iam doing in labeling, compared to my coworkers" (55% agreed, M=0.2, SD=1.348)
• "Iflabeling included game elements, the label results would be better" (50% agreed, M=0.4, SD=0.993)
• "Iflabeling included game elements it would be much more fun" (65% agreed, M=0.9, SD=0.999)
• "Iwouldnotlike itifotherswere able to seemylabeling progress on a leaderboard" (45% agreed, M=0.35, SD=1.27)
• "Using game elements at work makes a company seem less serious" (30% agreed, 55% disagreed, M=-0.65, SD=1.306)

Files\\Data collection and quality challenges for deep learning

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On
No	Web of science	0.0447	3			
				1	S	07/02/2022 23:17

Compared to traditional machine learning, there is less need for feature engineering, but more need for significant amounts of data. We thus go through stateof-the-art data collection techniques for machine learning. Then, we cover data validation and cleaning techniques for improving data quality. Even if the data is still problematic, hope is not lost, and we cover fair and robust training techniques for handling data bias and errors. We believe that the data management community is well poised to lead the research in these directions. The presenters have extensive experience in developing machine learning platforms and publishing papers in top-tier database, data mining, and machine learning venues.

				2	S	14/02/2022 14:12

Data collection for machine learning [13]. The techniques in the leaf nodes that are at least partially proposed by the data management community are highlighted in italic blue font. A key observation is that there is an convergence of techniques between the data management and machine learning communities, so one needs to know both sides to understand the overall research landscape.

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On
				3	S	07/02/2022 23:18

Noisy or Missing Labels. Regarding noisy labels, recent techniques are mainly categorized into loss correction and sample selection. The former estimates the confidence of a label for each sample and adjusts the loss for the sample based on its label confidence during backward propagation. The latter also estimates the confidence of a label for each sample and includes the samples in training only if their label confidence is above some threshold. Recently, the sample selection approach becomes dominant, and a hybrid of the two approaches has been proposed [15]. Regarding missing labels, semi-supervised learning builds a model from a mixture of labeled and unlabeled data, by adopting unsupervised loss or collaborating with mix-up augmentation for unlabeled data. The representative techniques will be selectively covered in this tutorial.
Missing Data. Because missing data can reduce the statistical power and produce biased estimates, data imputation has been an active

Files\\Towards Accountability for Machine Learning Datasets~ Practices from Software Engineering and Infrastructure

No	Scopus	0.0212	4			
				1	S	03/02/2022 10:47

Datasets that power machine learning are often used, shared, and reused with little visibility into the processes of deliberation that led to their creation. As artificial intelligence systems are increasingly used in high-stakes tasks, system development and deployment practices must be adapted to address the very real consequences of how model development data is constructed and used in practice. This includes greater transparency about data, and accountability for decisions made when developing it.

				2	S	03/02/2022 10:51

Despite rapid growth, the disciplines of data-driven decision making—including ML—have come under sustained criticism in recent years due to their tendency to perpetuate and amplify social inequality [13, 44]. Data is frequently identified as a key source of these failures through its role in “bias-laundering” [40, 51, 54, 119, 125]. For example, recent studies have uncovered widespread prevalence of undesirable biases in ML datasets, such as the underrepresentation of minoritized groups [27, 40, 131] and stereotype aligned correlations [28, 51, 72, 155]. Datasets also frequently reflect historical patterns of social injustices, which can subsequently be reproduced by ML systems built from the data. For example, in a recent study examining the datasets underlying predictive policing models deployed in police precincts across the US, the underlying data source was found to reflect racially discriminatory and corrupt policing practices [119]. The norms and standards of data collection within ML have themselves been subject to critique, with scholars identifying insufficient documentation and transparency regarding processes of dataset construction [52, 53, 126], as well as problematic consent practices [114]. The lack of accountability to datafied and surveilled populations as well as groups impacted by data-driven

				3	S	03/02/2022 10:52
				4	S	03/02/2022 10:55

Files\\Towards Building Robust DNN Applications~ An Industrial Case Study of Evolutionary Data Augmentation

No	IEEE	0.0399	3			
				1	S	11/02/2022 14:39

Data augmentation techniques that increase the amount of training data by adding realistic transformations are used in machine learning to improve the level of accuracy. Recent studies have demonstrated that data augmentation techniques improve the robustness of image classification models with open datasets; however, it has yet to be investigated whether these techniques are effective for industrial datasets. In this study, we investigate the feasibility of data augmentation techniques for industrial use.

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On
				2	S	11/02/2022 14:40

To improve the robustness of an ML model, a number of studies
have focused on data augmentation (DA) techniques [3, 4, 12, 17]. DA is a technique for providing data with realistic variations to ML models during the training phase. The variations in such transformations vary by domain. For example, photo images have variations such as an inversion, translation, rotation, zoom, occlusion, brightness, and contrast [4, 12]. In DA, these transformations are often applied randomly to a dataset (hereinafter referred to as the Random approach). Such techniques are implemented in major deep learning frameworks including PyTorch [8] and Keras [2]. Engstrom et al. showed that the 'Worst ofk' method outperforms the Random method in terms of improvement to the robustness of ML models [3].

				3	S	11/02/2022 14:41

methods perform well for open benchmark datasets, their performance in industrial systems has yet to be evaluated. Therefore, in this study, we investigate the effectiveness of the Worst ofk and Sensei approaches using our industrial graphical user interface (GUI) recognition system and determine the feasibility of these techniques in cases of real industrial use. We evaluate the DA techniques in a stepwise manner using image classification and object detection models because there are differences not only in the data domains but also in the target ML tasks between the existing studies and our proposed approach. The existing studies on Worst ofk and Sensei target image classification tasks using photographic images (e.g., an animal and vehicle image dataset [6] and a traffic sign image dataset [14]), whereas our GUI recognition system targets an object detection task using GUI screenshot images.

Nodes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\ML Model Engineering\concept drift

PDF

Files\\All versus one~ an empirical comparison on retrained and incremental machine learning for modeling performance of adaptable software

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On
No	Scopus	0.0134	2			
				1	S	10/02/2022 11:41

RQ4: How the modeling methods can be affected by the runtime fluctuations of the adaptable software, i.e., the number of concept drifts and the deviations in the data?
The errors of both modeling methods exhibit considerably positive monotonic correlations to the number of drifts, and non-trivial negative monotonic correlations to the deviations of data. We did not observe clear correlations of their training time to the number of concept drift and data deviations in general. The only exception is the strong correlation between training time of incremental modeling and the number of concept drift.

				2	S	10/02/2022 11:44

C. Analysis of the Fluctuation in Subject Software Systems To analyze the fluctuation of the adaptable software, we use
the following criteria to represents the changes at runtime: Concept Drift: the concept drift [46] refers to the statistical properties of the target performance indicator, which the model is trying to predict, change over time in unforeseen ways. In general, for real-world software and data as what we studied in this work, there is no exact understanding about when the concept drift occurs. Therefore, we leverage ADWIN [47], a well-known drift detector, to measure the number of drifts in the data stream. Since we can only count the number of drifts not the extents of drifts, we apply another metric below. Relative Standard Deviations (RSD): RSD measures the extents of change in the data stream by calculating the ratio between standard deviations and mean. This includes the data about the performance indicators and the related features of the software that can be used to train a model. The normalized nature of RSD allows us to report the mean value of the RSD, denoted as mRSD, for the features and performance indicators under all cases. A larger mRSD often imply that the overall extent of concept drifts is also more significant.

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On

Files\\An Empirical Study of the Impact of Data Splitting Decisions on the Performance of AIOps Solutions

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On
No	Google Scholar	0.0258	4			
				1	S	10/02/2022 10:59

• Concept drift: Over time, the distribution of the operation data and the relationship between the variables in the data may be constantly evolving [17, 49, 51] (a.k.a. concept drift [64, 84–86]). Concept drift may lead to obsolescence of the models trained on historical data, i.e., a model trained on outdated data may perform poorly on new data.

				Reference Number	Coded By Initials	Modified On
				2	S	10/02/2022 11:01

The Challenge of Concept Drift: In machine learning and data mining, the distribution of
the data and the relationship between the variables may evolve over time, which is known as concept drift [64, 84–86]. Concept drift may lead to obsolescence of models trained on previous data and negatively impact the performance. To mitigate the impact of concept drift, prior works propose approaches for detecting concept drift [26, 30, 64, 87] and handling concept drift [9, 12, 21, 28, 32, 60, 61, 77, 85]. For example, Nishida et al. [64] propose a concept drift detection method using statistical testing. It assumes that the prediction accuracy on the data from a recent time window would be equal to the overall accuracy if the target concept is stationary, and a significant decrease in the recent accuracy suggests a concept drift. When there is concept drift, aside from retraining a model from scratch, online learning updates the current model using the most recent data incrementally. Such model process input examples one-by-one and update

				Reference Number	Coded By Initials	Modified On
				3	S	10/02/2022 11:01

the model after receiving each example [28]. For example, CVFDT [33] is a decision tree model that incrementally updates itself when new data becomes available and can adapt to the drifting concept.
Time-based ensembles combine individual base models trained on data from small time peri-
ods. The intuition is that the base models trained from such small time periods can better capture the relationship between the variables, as the concept drift in a smaller period is relatively small. For example, Steet and Kim propose the Streaming Ensemble Algorithm (SEA) [77], which is a majority-voting ensemble approach that constantly replaces the weakest classifier in the ensemble with a quality measure that considers both the accuracy and diversity ofclassifiers in the ensemble. Cano and Krawczyk propose the Kappa Updated Ensemble (KUE) [12], which is a combination of online and block-based ensemble approaches. KUE uses the Kappa statistic for dynamic weighing and selection of base classifiers. Other advanced techniques in handling concept drift include an enhancement of the time-based ensemble methods by Krawczyk et al. [43] that improves the model's robustness to drift and noise by adding abstaining options to classifiers, allowing classifiers in the ensemble to refrain from making a decision if they have a confidence level below a specified threshold. Cano et al. [11]propose a rule-based classifier for drifting data streams using grammar-guided genetic programming. The model, namely, evolving rule-based classifier for drifting data streams (ERulesD2S), can provide accurate predictions and adapt to concept drift while offering the full interpretability based on classification rules.

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On
				4	S	24/02/2022 09:06

In machine learning and data mining, concept drift means the change in the relationships between the variables over time [84–86, 92]. Concept drift may negatively impact the performance of a model trained from the past data when applied to the new data [84–86]. Therefore, in this RQ, we analyze the studied datasets to understand whether concept drift issues exist in the context of AIOps. In particular, we leverage statistical analysis to measure the existence of concept drift in the studied datasets.
6.2 Approach Prior work [42, 45, 64] assumes that, given a stationary data distribution (i.e., no concept drift), a model trained on a previous time period would achieve a prediction performance (when evaluated on the next time period) that has no statistical difference from the prediction performance on the training period.We follow the same hypothesis to measure the concept drift in our studied datasets. If a model trained from the previous data shows a statistically significant performance difference on the new data, then a concept drift exists. In our study, we use the natural time intervals (i.e., one-day periods for the Google dataset
and one-month periods for the Backblaze dataset) to split the data into different time periods. We choose such a time window size as prior works have applied similar update strategies. For example, Lin et al. [51] update their model deployed in a production cloud service system with data from a one-month window. Similarly, Li et al. [49] consider retraining their model periodically and they also apply a one-month window. Also, Xu et al. [89] perform a daily model update with the data in a 90-day sliding window. We conduct our experiment as follows:
(1) For each time period, we train a model using the data from that time period and test the same model using the next time period's data to measure the prediction error rate.
(2) We then compute the statistical difference between the model's prediction error rate on the training time period and its prediction error rate on the testing time period, similar to prior work [45, 64]. However, these studies [45, 64] do not explicitly explain how they measure the prediction error rate on the training time period. Thus, we follow prior work [42, 86] and use 10-fold cross-validation on the training time period to measure the prediction error rate on the training time period.
(3) Similar to prior work [45, 64], we use a two-proportion Z-test to compute the statistical difference between the model's prediction error rates in the training and testing time periods, which is described as follows:
Z = p^(1 − ^p)
(^p2 − ^p1) − 0 1
n1 + 1 n2 , (1)
where p^1 is the prediction error rate in the training time period, p^2 is the prediction error rate in the testing time period, p^ is the overall prediction error rate, and n1 and n2 are the number of samples in the training time period and the testing time period, respectively.
(4) We then determine the significance level (i.e., p-value) from the Z-test. When the p-value is less than 0.05, we reject the null hypothesis (i.e.,

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On

Files\\Challenges in Deploying Machine Learning~ a Survey of Case Studies

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On
No	Google Scholar	0.0169	1			
				1	S	24/02/2022 10:54

6.3 Updating
Once the initial deployment of the model is completed, it is often necessary to be able to update the model later on in order to make sure it always reflects the most recent trends in data and the environment. There are multiple techniques for adapting models to a new data, including scheduled regular retraining and continual learning [55]. Nevertheless in production setting model updating is also affected by practical considerations.
A particularly important problem that directly impacts the quality and frequency of model update procedure is the concept drift. Concept drift in ML is understood as changes observed in joint distribution p(X, y), where X is the model input and y is the model output. Undetected, this phenomenon can have major adverse effects on model performance, as is shown by Jameel et al. [56] for classification problems or by Celik and Vanschoren [57] in AutoML context. Concept drift can arise due to a wide variety of reasons. For example, the finance industry faced turbulent changes as the financial crisis of 2008 was unfolding, and if advanced detection techniques were employed it could have provided additional insights into the ongoing crisis, as explained by Masegosa et al. [58]. Changes in data can also be caused by inability to avoid fluctuations in the data collection procedure, as described in paper Langenkämper et al. [59] which studies the effects of slight changes in marine images capturing gear and location on deep learning models' performance. Data shifts can have noticeable consequences even when occurring at microscopic scale, as Zenisek et al. [60] show in their research on predictive maintenance for wear and tear of industrial machinery. Even though concept drift has been known for decades [61], these examples show that it remains a critical problem for applications of ML today.
On top of the question of when to retrain the model to keep it up to date, there is an infrastructural question on how to deliver the model artifact to the production environment. In software engineering such tasks are commonly solved with continuous delivery (CD), which is an approach for
10
accelerating development cycle by building an automated pipeline for building, testing and deploying software changes. CD for machine learning solutions is complicated because, unlike in regular software products where changes only happen in the code, ML solutions experience change along three axis: the code, the model and the data. An attempt to formulate CD for ML as a separate discipline can be seen in Sato et al. [45]. This work describes the pieces involved and the tools that can be used at each step of building the full pipeline. A

Files\\Driftage~ a multi-agent system framework for concept drift detection

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On
No	Web of science	0.0321	3			
				1	S	07/02/2022 22:37

The amount of data and behavior changes in society happens at a swift pace in this interconnected world. Consequently, machine learning algorithms lose accuracy because they do not know these new patterns. This change in the data pattern is known as concept drift. There exist many approaches for dealing with these drifts. Usually, these methods are costly to implement because they require (i) knowledge of drift detection algorithms, (ii) software engineering strategies, and (iii) continuous maintenance concerning new drifts.

				2	S	07/02/2022 22:40

One approach to designing adaptive software is using the
MAPE-K (Monitor-Analyse-Plan-Execute over a shared Knowledge) software pattern for self-aware systems [27–30]. MAPE-K is organized into 4 components:
(i) The ”Monitor” is responsible for environmental monitoring, basically capturing data from sensors or what else the software knows about the environment and stores on the knowledge base (KB);
(ii) The ”Analyser” will enrich knowledge using the collected data from the environment and reporting to the KB the result of its analysis;
(iii) The ”Planner” understands the analysis made by analysers and makes decisions on it while saving this information into the KB; and
(iv) The ”Executor” gets decisions from the KB and knows how to execute them. The most

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On
				3	S	07/02/2022 22:40

Driftage is a modular framework based on MAPE-K, chosen as the pattern to model this agent-based framework because CDD needs high adaptability and fits very well with MAS. Each agent type in Driftage has only 1 accountable agent on
the MAPE-K architecture. Each agent can be implemented to follow the selected goal without affecting the others but can exchange information with others. Instead of an agent using the MAPE-K software pattern, an agent on the Driftage framework can be implemented following 1 of the 4 types: Monitor, Analyser, Planner, or Executor. Each type can generate multiple autonomous agents.
There are 2 main flows on this framework:
(i) Monitor–Analyser: for capture and fast prediction ofconcept drifts on data;
(ii) Planner–Executor: to analyse whether concept drift detected should be alerted.
These 2 flows can intercommunicate by means ofaKB, where
drifts are stored, and we make all history about drift analysis persistent. Each agent communicates through an XMPP server on the framework because the implementation extends Spade [49], which is a library for MAS using Python. The XMPP protocol solves some problems with MAS, already providing authentication and communication channels for the agents. XMPP servers also work for load balancing and guarantee message exchanges. We have implemented Driftage using Python because data
engineers widely use it, and it enables the programmer to answer the system's requirements

Nodes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\ML Model Engineering\HPO

PDF

Files\\Auptimizer -- an Extensible, Open-Source Framework for Hyperparameter Tuning

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On
No	Web of science	0.0156	2			
				1	S	08/02/2022 13:46

Tuning machine learning models at scale, especially
finding the right hyperparameter values, can be difficult and time-consuming. In addition to the computational effort required, this process also requires some ancillary efforts including engineering tasks (e.g., job scheduling) as well as more mundane tasks (e.g., keeping track of the various parameters and associated results).

				2	S	08/02/2022 13:48

There is no universal HPO algorithm having the best performance over all problems. Thus, trying different ones is necessary to reveal the best results and business value. However, a high adoption cost commonly prevents user from trying different algorithms. We summarize the common factors that limit the current HPO toolboxes as flexibility, usability, scalability, and extensibility: • Flexibility. It is challenging to switch between HPO algorithms, as the interfaces are dramatically different.
• Usability. It is time-consuming to integrate an existing ML project into an HPO package. Often, users need to rewrite their code for a specific HPO toolbox, and resulting script cannot be used anywhere else.
• Scalability. The integration with large-scale computational resources is missing and it is typically hard to scale the toolbox to a multi-node environment.
• Extensibility. It is challenging to introduce a new algorithm into the existing libraries as these libraries are tightly coupled with the implemented algorithms.
We summarize the comparison of representative HPO so-
lutions based on the above criteria in Table I. Based on our experience in developing an in-house solution, we release an HPO framework, Auptimizer, to mitigate the above-mentioned challenges.

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On

Files\\HyperNOMAD~ Hyperparameter Optimization of Deep Neural Networks Using Mesh Adaptive Direct Search

No	ACM Digital library	0.0206	1			
				1	S	07/02/2022 16:20

However, the performance of a neural network is strongly linked to its structure and to the
values of the parameters of the optimization algorithm used to minimize the error between the predictions of the network and the data during its training. The choices of the neural network hyperparameters can greatly affect its ability to learn from the training data and to generalize with new data. The algorithmic hyperparameters of the optimizer must be chosen a priori and cannot be modified during optimization. Hence, to obtain a neural network, it is necessary to fix several hyperparameters of various types: real, integer, and categorical. A variable is categorical when it describes a class, or category, without a relation of order between these categories. The search for an optimal configuration is a very slow process that, along with the training, takes up the majority of the time when developing a network for a new application. It is a relatively new problem that is often solved randomly or empirically. Derivative-free optimization (DFO) [8, 21] is the field that aims to solve optimization prob-
lems where derivatives are unavailable, although they might exist. This is the case, for example, when the objective and/or constraint functions are non-differentiable, noisy, or expensive to evaluate. In addition, the evaluation in some points may fail, especially if the values of the objective and/or constraints are the outputs of a simulation or an experience. Blackbox optimization (BBO) is a subfield of DFO where the derivatives do not exist and the problem is modeled as a blackbox. This term refers to the fact that the computing process behind the output values is unknown. The general DFO problem is described as
min $x \in \Omega$
$f(x)$,
where f is the objective function to minimize over the domain Ω. There are two main classes ofDFO methods: model-based and direct search methods. The first
uses the value of the objective and/or the constraints at some already evaluated points to build a model able to guide the optimization by relying on the predictions of the model. For example, this class includes methods based on trust regions [21, Chapter 10] or interpolation models [52]. This differentiates them from direct search methods [31] that adopt a more straightforward strategy to optimize the blackbox. At each iteration, direct search methods generate a set of trial points that are compared to the "best solution" available. For example, the GPS algorithm [59]definesamesh on the search space and determines the next point to evaluate by choosing a search direction. DFO algorithms usually include a proof ofconvergence that ensures a good-quality solution under certain hypotheses on the objective function. BBO algorithms extend beyond this scope by including heuristics such as evolutionary algorithms, sampling methods, and so on. In [5, 10], the authors explain how a hyperparameter optimization (HPO) problem can be
seen as a blackbox optimization problem. Indeed, the HPO problem is equivalent to a blackbox that takes the hyperparameters of a given algorithm and returns some measure of performance defined in advance such as the time to solution, the value of the best point found, or the number of solved problems. In the case of neural networks, the blackbox can return the accuracy on the test dataset as a measure

Files\\Tunability~ Importance of Hyperparameters of Machine Learning Algorithms

No	Scopus	0.0272	3			
				1	S	11/02/2022 14:54

In order to select an appropriate hyperparameter configuration for a specific dataset
at hand, users of ML algorithms can resort to default values of hyperparameters that are specified in implementing software packages or manually configure them, for example, based on recommendations from the literature, experience or trial-and-error.

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On
				2	S	11/02/2022 14:54

Alternatively, one can use hyperparameter tuning strategies, which are data-dependent, second-level optimization procedures (Guyon et al., 2010), which try to minimize the expected generalization error of the inducing algorithm over a hyperparameter search space of considered candidate configurations, usually by evaluating predictions on an independent test set, or by running a resampling scheme such as cross-validation (Bischl et al., 2012). For a recent overview of tuning strategies, see, e.g., Luo (2016). These search strategies range from simple grid or random search (Bergstra and Bengio, 2012) to more complex, iterative procedures such as Bayesian optimization (Hutter et al., 2011; Snoek et al., 2012; Bischl et al., 2017b) or iterated F-racing (Birattari et al., 2010; Lang et al., 2017). In addition to selecting an efficient tuning strategy, the set of tunable hyperparameters and their corresponding ranges, scales and potential prior distributions for subsequent sampling have to be determined by the user. Some hyperparameters might be safely set to default values, if they work well across many different scenarios. Wrong decisions in these areas can inhibit either the quality of the resulting model or at the very least the efficiency and fast convergence of the tuning procedure. This creates a burden for:
1. ML users—Which hyperparameters should be tuned and in which ranges? 2. Designers of ML algorithms—How do I define robust defaults?
We argue that many users, especially if they do not have years of practical experience in the field, here often rely on heuristics or spurious knowledge. It should also be noted that designers of fully automated tuning frameworks face at least very similar problems. It is not clear how these questions should be addressed in a data-dependent, automated, optimal and objective manner. In other words, the scientific community not only misses answers to these questions for many algorithms but also a systematic framework, methods and criteria,

				3	S	11/02/2022 14:56

Our study has some limitations that could be addressed in the future: a) We only considered binary classification, where we tried to include a wider variety of datasets from different domains. In principle this is not a restriction as our methods can easily be applied to multiclass classification, regression, survival analysis or even algorithms not from machine learning whose empirical performance is reliably measurable on a problem instance. b) Uniform random sampling of hyperparameters might not scale enough for very high dimensional spaces, and a smarter sequential technique might be in order here, see Bossek et al. (2015) for an potential approach of sampling across problem instances to learn optimal mappings from problem characteristics to algorithm configurations. c) We currently are learning static defaults, which cannot depend on dataset characteristics (like number of features, or further statistical measures). Doing so might improve performance results of optimal defaults considerably, but would require a more complicated approach. A recent paper regarding this topic was published by van Rijn et al. (2018). d) Our approach still needs initial ranges to be set, in order to run our sampling procedure. Only based on these wider ranges we can then

Files\\Ultron-AutoML~ an open-source, distributed, scalable framework for efficient hyper-parameter optimization

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On
No	IEEE	0.0188	1			
				1	S	03/02/2022 10:09

Hyperparameter Optimization (HPO), also referred to as AutoML in the literature, can be cast as the optimization of an unknown, possibly stochastic, objective function mapping the hyper-parameter search space to a real valued scalar, the ML model's accuracy or any other performance metric on the validation dataset. The search-space can extend beyond algorithm or architecture specific elements to encompass the space of data pre-processing and data-augmentation techniques, feature selections, as well as choice of algorithms. This is sometimes referred to as the CASH (Combined Algorithm Search and Hyper-parameter tuning) problem for which algorithms have been proposed [28], [48]. Neural Architecture Search (NAS) is a special type of HPO where the focus is on algorithm driven design of neural network architecture components or cells [26]. Models trained with architectures composed of these algorithmically designed neural network cells have been shown to outperform their hand-crafted counterparts in image recognition, object detection [57], and semantic segmentation [21], underscoring the practical importance of this field. Random Search [18] and Grid Search are effective HPO strategies when the computational budget is limited or the hyper-parameter search space is high dimensional. Both are easy to implement and completely parallelizable. Random Search is also widely regarded as a good baseline for benchmarking new hyper-parameter optimization algorithms [33]. Bayesian Optimization (BO) is a dominant paradigm for HPO [20], [27], [45]. Here, the objective function is modeled as a Gaussian Process [50], with the Kernel design reflecting assumptions about the objective function's smoothness properties. Under this assumption, the posterior distribution of the validation score for a candidate architecture is a Gaussian

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On

Nodes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\ML Model Engineering\ML monitoring

PDF

Files\\AI lifecycle models need to be revised

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On
No	Google Scholar	0.0019	1			
				1	S	24/02/2022 13:09

4.8 Model Monitoring
After having a model in production, it is necessary to keep track of its behavior to make sure it operates as expected. It implies testing the model while the model is deployed online. The main advantage is that it uses real data. Previous work refers to this stage as online testing (Zhang et al. 2020).

Files\\Challenges in Deploying Machine Learning~ a Survey of Case Studies

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On
No	Google Scholar	0.0161	1			
				1	S	24/02/2022 10:50

Monitoring is one of the issues associated with maintaining machine learning systems as reported by Sculley et al. [52]. The community is in the early stages of understanding what are the key metrics of data and models to monitor and how to alarm on them. Monitoring of evolving input data, prediction bias and overall performance of ML models is an open problem. Another maintenance issue highlighted by this paper that is specific to data-driven decision making is feedback loops. ML models in production can influence their own behavior over time via regular retraining. While making sure the model stays up to date, it is possible to create feedback loop where the input to the model is being adjusted to influence its behavior. This can be done intentionally, as well as happen inadvertently which is a unique challenge when running live ML systems.
Klaise et al. [53] point out the importance of outlier detection as a key instrument to flag model predictions that cannot be used in a production setting. The authors name two reasons for such predictions to occur: the inability of the models to generalize outside of the training dataset and also overconfident predictions on out-of-distribution instances due to poor calibration. Deployment of the outlier detector can be a challenge in its own right, because labeled outlier data is scarce, and the detector training often becomes a semi-supervised or even an unsupervised problem. Additional insight on monitoring of ML systems can be found in Ackermann et al. [54]. This paper describes an early intervention system (EIS) for two police departments in the US. On the surface their monitoring objectives seem completely standard: data integrity checks, anomaly detection and performance metrics. One would expect to be able to use out-of-the-box tooling for these tasks. However, the authors explain that they had to build all these checks from scratch in order to maintain good model performance. For instance, the data integrity check meant verifying updates of a certain input table and checksums on historical records, performance metric was defined in terms of the number of changes in top k outputs, and anomalies were tracked on rank-order correlations over time. All of these monitoring tools required considerable investigation and implementation. This experience report highlights a common problem with currently available end-to-end ML platforms: the final ML solutions are usually so sensitive to problem's specifics that out-of-the-box tooling does not fit their needs well.
As a final remark we note that there is an overlap between choice of metrics for monitoring and validation. The latter topic is discussed in

Files\\Overton~ A Data System for Monitoring and Improving Machine-Learned Products

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On
No	Google Scholar	0.0050	1			
				1	S	04/02/2022 21:53

Fine-grained Quality Monitoring While overall improvements to quality scores are important, often the weekto-week battle is improving fine-grained quality for important subsets of the input data. An individual subset may be rare but are nonetheless important, e.g., 0.1% of queries may correspond to a product feature that appears in an advertisement and so has an outsized importance. Traditional machine learning approaches effectively optimize

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On

Files\\Software Logs for Machine Learning in a DevOps Environment

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On
No	Scopus	0.0509	5			
				1	S	11/02/2022 14:25

System logs perform a critical function in software-
intensive systems as logs record the state of the system and significant events in the system at important points in time. Unfortunately, log entries are typically created in an ad-hoc, unstructured and uncoordinated fashion, limiting their usefulness for analytics and machine

				2	S	11/02/2022 14:26

In a DevOps environment, especially, unmanaged evolution in log data structure causes frequent disruption of operations in automated data pipelines, dashboards and analytics.

				3	S	11/02/2022 14:27

Our research shows that source code is often not available for analysis and many logs are highly unstructured and consequently difficult to parse. The problems concerning access to source code [9] and
lack of structure [4] have been acknowledged in research and automated log parsers, such as MoLFI [9], Drain [7], and Spell [5], have been developed to some success. However, even the state-of-the-art log parser, Drain [7], struggles with state identification and dealing with log messages of variable length, which leads to varying and unpredictable performance based on the type of log [18]. In practice, many companies have logs with less standardization and automated log parsing does not provide the desired results. Finally, some logging standards have been proposed but
these are often domain specific or insufficient. For example, the XES standard [6] is simple to parse, but the transformation

				4	S	11/02/2022 14:27

Although logging may seem trivial, in practice most R&D
teams aim to manually observe the functionality of their systems. This leads to a high degree of variance in log generation, multiple log files for generating log entries of different types, and, in a DevOps environment, continuous and unmanaged changes to internal logging practices. For ML, this can greatly complicate processing the data and hinder the training process. The traditional way of generating system logs is shown
in figure 1. In this case, the R&D teams have full freedom to generate logs to optimally support their needs. These approaches often have developed over time to optimally support developers. The challenge is that when the same logs are used for machine learning, the data science team is required to spend significant effort on pre-processing, the pre-processed data then used to manually (re)train the model which is, subsequently, manually deployed. As part of our case study research at the primary and secondary case companies as well as based on the literature that we reviewed and reported in section II, we have identified eight significant challenges associated with

				5	S	11/02/2022 14:27

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On

Files\\Towards MLOps~ A Framework and Maturity Model

No	Scopus	0.0246	1			
				1	S	24/02/2022 10:37

C. Semi-automated Model Monitoring: At this stage, companies have a manual model monitoring in place. With MLOps, they can attain a transition from manual monitoring to semi-automated model monitoring. Preconditions: To reach this transition, there should be provisions for triggering [43] when performance degrades and availability of tools for diagnostics, performance monitoring and addressing model drift [43] [27] [36]. It also requires
338 Authorized licensed use limited to: UNIVERSITY OF OSLO. Downloaded on January 31,2022 at 12:51:22 UTC from IEEE Xplore. Restrictions apply.
automation scripts to manage and monitor models based on drift [38] and ability to perform continuous model tracking [31]. For easy monitoring of models, MLOps professionals has to be provided with visual tools [34], and dedicated and centralized dashboards [38] [27] [28]. It also requires data orchestration pipelines and rule-based data governance to ensure data changes [31], feedback loop and continuous model retraining [43]. There should be also a mechanism to automatically train model in production using fresh data based on live pipeline triggers and feedback loops [38] D. Fully-automated Model Monitoring: The companies have deployment and monitoring of models in place where performance degradation is acknowledged by alert. By utilizing MLOps, they undergo transition towards fully automated monitoring of models. Preconditions: For this transition, company requires CI/CD integration with automation and orchestration [43] and CT pipeline to retrain models when performance degrades [31]. For this transition, there is a need to ensure certification of models [32] [23], governance and security controls [43] [34] [36], model explainability [43] [36], auditing of model usage [34] [43], reproducible workflow and models [36]. There should be mechanisms to perform end-to-end QA test and performance checks [43]. There should be assurance that data security and privacy requirements are built into data pipelines [31] as well as retrain production

Nodes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\ML Model Engineering\Model deployment

PDF

Files\\A comprehensive study on challenges in deploying deep learning based software

No	ACM Digital library	0.0923	14			
				1	S	11/02/2022 13:20

Deep learning (DL) becomes increasingly pervasive, being used in a wide range of software applications. These software applications, named as DL based software (in short as DL software), integrate DL models trained using a large data corpus with DL programs written based on DL frameworks such as TensorFlow and Keras. A DL program encodes the network structure of a desirable DL model and the process by which the model is trained using the training data. To help developers of DL software meet the new challenges posed by DL, enormous research efforts in software engineering have been devoted. Existing studies focus on the development of DL software and extensively analyze faults in DL programs. However, the deployment of DL software has not been comprehensively studied.

				2	S	24/02/2022 10:28

DL software deployment. After DL software has been well
validated and tested, it is ready to be deployed to different platforms for real usage. The deployment process focuses on platform adaptations, i.e., adapting DL software for the deployment platform. The most popular way is to deploy DL software on the server or cloud platforms [107]. This way enables developers to invoke services powered by DL techniques via simply calling an API endpoint. Some frameworks (e.g., TF Serving [68]) and platforms (e.g., Google Cloud ML Engine [61]) can facilitate this deployment. In addition, there is a rising demand in deploying DL software to mobile devices [102] and browsers [91]. For mobile platforms, due to their limited computing power, memory size, and energy capacity, models that are trained on PC platforms and used in the DL software cannot be deployed directly to the mobile platforms in some cases. Therefore, some lightweight DL frameworks, such as TF Lite for Android and Core ML for iOS, are specifically designed for converting pre-trained DL models to the formats supported by mobile platforms. In addition, it is a common practice to perform model quantization before deploying DL models to mobile devices, in order to reduce memory cost and computing overhead [83, 102]. For model quantization, TF Lite supports only converting model weights from floating points to 8-bit integers, while Core ML allows flexible quantization modes, such as 32 bits to 16/8/4 bits [83]. For

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On
				3	S	11/02/2022 13:24
				4	S	11/02/2022 13:25

Figure 3 shows the popularity trend of deploying DL software in terms of the number of users and questions on SO. The figure indicates that this topic is gaining increasing attention, demonstrating the timeliness and urgency of this study. For deploying DL software on server/cloud platforms, we observe
that users and questions increase in a steady trend. In 2017, most major vendors roll out their DL frameworks for mobile devices [102]. As a result, we can observe that both the number of users and the number of questions related to mobile deployment in 2017 increase by more than 300% compared to 2016. For deploying DL software on browsers, questions start to appear in 2018 due to the release of TF.js in 2018. As found by Ma et al. [91], DL in browsers is still at

				5	S	11/02/2022 13:25

RQ2: DIFFICULTY For deployment and other aspects (in short of non-deployment) of DL software, the percentages of relevant questions with no accepted answer (%no acc.) are 70.7% and 62.7%, respectively. The significance of this difference is ensured by the result of proportion test (⍰2 = 78.153, df = 1, ⍰-value < 2.2e-16), indicating that questions related to DL software deployment are more difficult to answer than those related to other aspects of DL software. More specifically, for server/cloud, mobile, and browser deployment, the values of%no acc. are 69.8%, 71.6%, and 69.1%, respectively. In terms of this metric, questions about deploying DL software are also more difficult to resolve than other well-studied challenging topics in SE, such as big data (%no acc. = 60.5% [75]), concurrency (%no acc. = 43.8% [72]), and mobile (%no acc. = 55.0% [96]). Figure 4 presents the boxplot of response time needed to receive
an accepted answer for deployment and non-deployment related questions.We can observe that the time needed for non-deployment questions is mostly concentrated below 600 minutes, while

				6	S	11/02/2022 13:26

6.1 Common Challenges in Server/Cloud, Mobile, and Browser
To avoid duplicate descriptions, we first present the common inner categories in Server/Cloud, Mobile, and Browser. 6.1.1 General Questions. This category shows general challenges that do not involve a specific step in the deployment process, and contains several leaf categories as follows. Entire procedure of deployment. This category refers to gen-
eral questions about the entire procedure of deployment, mainly raised without practical attempts. These questions are mainly in the form of “how”, such as “how can I use that model in android for image classification” [6]. In such questions, developers often complain about the documentation, e.g., “there is no documentation given for this model” [7]. Answerers mainly handle these questions by providing existing tutorials or documentation-like information that does not appear elsewhere, or translate the jargon-heavy documentation into case-specific guidance phrased in a developer-friendly way. Compared to Server/Cloud (9.7%) and Mobile (13.4%), Browser contains relatively fewer such questions (3.2%). A possible explanation is that since DL in browsers is still in the early stage [91], developers are mainly stuck in DL’s primary usage rather than being eager to explore how to apply DL to various scenarios. Conceptual questions. This category includes questions about
basic concepts or background knowledge related to DL software deployment, such as “is there any difference between these Neural

				7	S	11/02/2022 13:26
				9	S	11/02/2022 13:27

Limitations ofplatforms/frameworks. This category is about limitations of relevant platforms or DL frameworks. For example, a senior software engineer working on the Google Cloud ML Platform team apologizes for the failure that a developer encounters, admitting that the platform currently does not support batch prediction [9]. Besides, some issues reflect bugs in current deployment related frameworks. For instance, an issue reveals a bug in the TocoConvert.from_keras_model_file method of TF Lite [10].
6.1.2 Model Export andModel Conversion. Both categories cover challenges in converting DL models in DL software into the formats supported by deployment platforms. Model export directly saves the trained model into the expected format, and it is a common way for deploying DL models

				10	S	11/02/2022 13:27

6.2 Common Challenges in Mobile and Browser
6.2.1 Data Extraction. To deploy DL software successfully, developers need to consider any stage that may affect the final performance, including data extraction. This category is observed only in Mobile and Browser, accounting for 1.7% and 3.2% of questions, respectively. This finding indicates the difficulty of extracting data in mobile devices and browsers.
6.2.2 Inference Speed. Compared to server/cloud platforms, mobile and browser platforms have weaker computing power. As a result, the inference speed of the deployed software has been a challenge in mobile devices (3.9%) and browsers (7.2%).

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On
				11	S	11/02/2022 13:27

6.3 Common Challenges in Server/Cloud and Browser
Environment. This category includes challenges in setting up the environment for DL software deployment, and accounts for 19.4% and 19.2% of questions in Server/Cloud and Browser, respectively. For Mobile, its environment related questions are mainly distributed in DL Library Compilation and DL Integration into Projects categories that will be introduced later. When deploying DL software to server/cloud platforms, developers need to configure various environment variables, whose diverse options make the configuration task challenging. In addition, for the server deployment, developers also need to install or build necessary frameworks such as TF Serving. Issues that occur in this phase are included in Installing/building frameworks. Similarly, when deploying DL software

				12	S	11/02/2022 13:27

6.4 Remaining Challenges in Server/Cloud 6.4.1 Request. This category covers challenges in making requests in the client and accounts for 13.3% of questions in Server/Cloud. For Request, developers have difficulty in configuring the request body [34], sending multiple requests at a single time (i.e., batching request) [35], getting information ofserving models via request [36], etc.

				13	S	11/02/2022 13:27

6.5 Remaining Challenges in Mobile 6.5.1 DL Library Compilation. This category includes challenges in compiling DL libraries for target mobile devices and covers 7.8% of questions in Mobile. Since Core ML is well supported by iOS, developers can use Core ML directly without installing or building it. For TF Lite, pre-built libraries are officially provided for developers' convenience. However, developers still need to compile TF Lite from source code by themselves in some cases (e.g., deploying models containing unsupported operators). Since the operators supported by TF Lite are still insufficient to meet developers' demand [43], developers sometimes need to register unsupported operators manually to add them into the run-time library. It may be challenging for developers who are unfamiliar with TF Lite. In addition, for compilation, developers need to configure build command lines and edit configuration files (i.e., Build configuration). Wrong configurations [44] can result in build failure or library incompatibility with target platforms.

				14	S	11/02/2022 13:27

6.6 Remaining Challenges in Browser Model Loading. This category includes challenges in loading DL models in browsers, being the most common challenges in browser deployment (accounting for 24.0% of questions). For browsers, TF.js provides a tf.loadLayersModel method to support loading models from local storage, Http endpoints, and IndexedDB. Among the three ways, we observe that the main challenge lies in loading from local storage (8.0%). In the official document of TF.js [50], "local storage" refers to the browser's local storage, which is interpreted in a hyperlink [51] contained in the document as that "the stored data is saved across browser sessions."

				15	S	11/02/2022 13:27

6.7 Unclear Questions
Although unclear questions are not included in our taxonomy, we also manually examine them to seek for some insights. All unclear questions have no accepted answers and do not have informative discussions or question descriptions to help us determine the challenges behind the questions. Among these unclear questions, 53% report unexpected results [54] or errors [55] when making predictions using the deployed models. However, no anomalies occur at any phase before the phase of making predictions, making it rather difficult to discover the underlying challenges. In fact, various issues can result in the errors or unexpected results in this phase.

Files\\AI lifecycle models need to be revised

No	Google Scholar	0.0091	1			
				1	S	24/02/2022 13:11

4.7 Model Deployment We observed three deployment patterns at ING:
1. A specialized team creates a prototype with a validated methodology, and an engineering team takes care of reimplementing it in a scalable, ready-to-deploy fashion. In some cases, this is a necessity due to the technical requirements of the model, e.g., when models are developed in Python, but should be deployed in Java (P08, P09, P13).
2. A specialized team creates a model and exports its configuration (e.g., a pickle9 and required dependencies) to a system that will semi-automatically bundle it and deploy it without changing the model (P01, P09).
3. The same team takes care of creating the model and taking it into production. This mostly means that software engineers are part of the team and a structured and strict software architecture is ensured.
Similar to the training environments, Machine Learning systems are deployed to on-
premises environments. A reported challenge regarding the deployment environment is that different hardware and platform parameters (e.g., Spark parameters) can result in different model behavior or errors (P16). For example, the deployment environment may have less memory than the training environment. Furthermore, the resources for a Machine Learning system are dynamically allocated whenever needed. However, it is not trivial understanding when a system is no longer needed and should be scaled down to zero (P01).
There are deploymentpatternsinwhicha separate team needs to reimplementthe model to meet production settings.

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On

Files\\An Empirical Study on Deployment Faults of Deep Learning Based Mobile Applications

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On
No	ACM Digital library	0.0161	2			
				1	S	10/02/2022 10:48

Deep learning (DL) is moving its step into a growing
number of mobile software applications. These software applications, named as DL based mobile applications (abbreviated as mobile DL apps) integrate DL models trained using large-scale data with DL programs. A DL program encodes the structure of a desirable DL model and the process by which the model is trained using training data. Due to the increasing dependency of current mobile apps on DL, software engineering (SE) for mobile DL apps has become important. However, existing efforts in SE research community mainly focus on the development of DL models and extensively analyze faults in DL programs. In contrast, faults related to the deployment of DL models on mobile devices (named as deployment faults of mobile DL apps) have not been well studied. Since mobile DL apps have been used by billions of end users daily for various purposes including for safety-critical scenarios, characterizing their deployment faults is of enormous

				Reference Number	Coded By Initials	Modified On
				2	S	10/02/2022 10:51

Recently, the rapid growth of mobile DL apps [22] has
posed urgent challenges to the deployment of DL models, i.e., deploying DL models on mobile devices. For example, computation-intensive DL models can be executed efficiently on PC/server platforms, but they cannot be directly deployed and executed on mobile devices with limited computing power [23]. Although major vendors have rolled out specific DL frameworks such as TF Lite [24] and Core ML [25] to facilitate this deployment process, various specific faults are still emerging in this process and frequently asked on Stack Overflow (SO), one of the most popular Q&A forums for developers [13]. Moreover, previous work [13] has demonstrated that relevant questions are increasing rapidly on SO and more difficult to resolve than those related to other aspects of DL based applications. In addition, mobile DL apps are not only used by billions of end users for their daily activities (e.g., speech-to-text and photo beauty) [22], [26], but also reported to be increasingly adopted in various safety-critical scenarios (e.g., driver assistance [27] and autonomous vehicles [28]). Therefore, the emerging faults related to the deployment of DL models on mobile devices (named as deployment faults of mobile DL apps) should be carefully addressed. Unfortunately, the characteristics of these faults have not been well understood.

Files\\Software engineering for artificial intelligence and machine learning software~ A systematic literature review

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On
No	Google Scholar	0.0020	1			
				1	S	23/02/2022 19:46

4.4.9. Model deployment Challenges regarding the deployment of the ML model in real or test environments
involve dependency management, maintaining the glue code, monitoring and logging, and the unintended feedback loops. For the deployment process, when deploying the trained models from a testing environment to an operating one, there lacks a benchmarking understanding of the migration and quantization processes, such as the impacts on prediction accuracy and performance (Guo et al., 2019). Relating to the deployment process, changing hardware and software, issues to maintain reproducible results, incur engineering costs for keeping software and hardware up to date (Munappy et al., 2019).

Files\\Towards MLOps~ A Framework and Maturity Model

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On
No	Scopus	0.0080	1			
				1	S	03/02/2022 10:24

To release ML models, package [41], validate [41] and
deploy models [40] to production [41]. When deploying a model to production, it has to be integrated with other models as well as existing applications [30] [41]. When the model is in production, it serves requests. Despite the fact that training is often a batch process, the inferences can be REST endpoint/custom code, streaming engine, micro-batch, etc. [35]. When performance drops, monitor the model [41] and enable the data feedback loop [41] to retrain the models . In a fully mature MLOps context, perform continuous integration and delivery by enabling the CI/CD pipeline and continuous retraining through CT pipeline [41] [31].

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On

Files\\Why is Developing Machine Learning Applications Challenging~ A Study on Stack Overflow Posts

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On
No	Web of science	0.0016	1			
				1	S	31/01/2022 14:43

(3) the data preprocessing and model deployment phases are where most of the challenges lay; and (4) addressing most of these challenges require more ML implementation knowledge than ML conceptual knowledge.

Nodes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\ML Model Engineering\Model development

PDF

Files\\Challenges in Deploying Machine Learning~ a Survey of Case Studies

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On
No	Google Scholar	0.0177	3			
				1	S	07/02/2022 23:36

In recent years, machine learning has received increased interest both as an academic research field and as a solution for real-world business problems. However, the deployment of machine learning models in production systems can present a number of issues and

2	S	07/02/2022 23:37

This shift comes with challenges. Just as with any other field, there are significant differences between what works in academic setting and what is required by a real world system. Certain bottlenecks and invalidated assumptions should always be expected in the course of that process. As more solutions are developed and deployed, practitioners sometimes report their experience in various forms, including publications and blog posts

3	S	07/02/2022 23:40

Files\\On testing machine learning programs

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On
No	Web of science	0.0026	1			
				1	S	07/02/2022 10:20

Model Engineering: Challenges and Issues Once ML engineers have collected and processed the
data, they proceed to finding the appropriate statistical learning model that could fit the available data in order to build its own logic and solve the given problem. A wide range of statistical models can be acquired and–or extended to suit different classification and regression purposes. There are simple models that make initial assumptions

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On

Files\\Software engineering for artificial intelligence and machine learning software~ A systematic literature review

No	Google Scholar	0.0031	1			
				1	S	23/02/2022 19:45

Model development The list of the challenges regarding the building of the ML model is related to different
aspects, such as a large number of model inputs, the AI ethics implementation, formal representation of complex models, optimization of feature engineering, imperfection and accuracy assurance, invalidation of models, model localization with data constraints, module documentation, uncertainty in input-output relationships, and uncertainty in model behaviour. In model development, the main challenge is to obtain a large number of model inputs
(Ishikawa, 2019; Renggli et al., 2019). It is difficult to clearly define the correction criteria for system outputs or correct outputs for each individual model input. Furthermore, for systems with a supervised learning paradigm, it is difficult to obtain labelled data that will serve as an
22
input for the model, mainly when there is a large volume of unlabelled data. For a supervised learning paradigm, all samples must be

Files\\Towards MLOps~ A Framework and Maturity Model

No	Scopus	0.0054	1			
				1	S	03/02/2022 10:24

In ML model development, provisions should be made to
run experiments in parallel, optimize the chosen model with hyperparameters, and finally evaluate the model to ensure that it fits the business case. After versioning, the code is stored in the code repository [42] [23]. The model repository [39] keeps track of the models that will be used in production, and the metadata repository contains all the information about the models (e.g., hyperparameter

Files\\Why is Developing Machine Learning Applications Challenging~ A Study on Stack Overflow Posts

No	Web of science	0.0060	1			
				1	S	23/02/2022 20:03

Model Fitting (MF) We assume the developer has a specific model in mind (e.g., SVM), so questions related to a specific model implementation, training, convergence determination, etc.
Model Tuning (MT) We assume the developer has trained a specific model and is aiming to fine tune it through hyper-parameter tuning, learning rate, regularization, etc.
Model Evaluation and Result Interpretation (ME)
Model Deployment and Environment Setup (MD)
Others
We assume the developer completed the training and tuning of a single or multiple ML models. Questions related to evaluation or measuring the performance of a model. Questions related to results interpretation
Questions related to environment setup, memory or storage issues, deployment performance tuning, etc

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On

Nodes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\ML Model Engineering\Model publishing and serving

PDF

Files\\Challenges in Deploying Machine Learning~ a Survey of Case Studies

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On
No	Google Scholar	0.0262	1			
				1	S	24/02/2022 11:04

6 Model Deployment
Machine learning systems running in production are complex software systems that have to be maintained over time. This presents developers with another set of challenges, some of which are shared with running regular software services, and some are unique to ML. There is a separate discipline in engineering, called DevOps, that focuses on techniques and tools required to successfully maintain and support existing production systems. Consequently, there is a necessity to apply DevOps principles to ML systems. However, even though some of the DevOps principles apply directly, there is also a number of challenges unique to productionizing machine learning. This is discussed in detail by Dang et al. [50] which uses the term AIOps for DevOps tasks for ML systems. Some of the challenges mentioned include lack of high quality telemetry data as well as no standard way to collect it, difficulty in acquiring labels which makes supervised learning approaches inapplicable3 and lack of agreed best practices around handling of machine learning models. In this section, we discuss issues concerning three steps within model deployment: integration, monitoring and updating.
6.1 Integration
The model integration step constitutes of two main activities: building the infrastructure to run the model and implementing the model itself in a form that can be consumed and supported. While the former is a topic that belongs almost entirely in systems engineering and therefore lies out of scope of this work, the latter is of interest for our study, as it exposes important aspects at the intersection of ML and software engineering. In fact, many concepts that are routinely used in software engineering are now being reinvented in the ML context. Code reuse is a common topic in software engineering, and ML can benefit from adopting the same mindset. Reuse of data and models can directly translate into savings in terms of time, effort or infrastructure. An illustrative case is the approach Pinterest took towards learning image embeddings [51]. There are three models used in Pinterest internally which use similar embeddings, and initially they were maintained completely separately, in order to make it possible to iterate on the models individually. However, this created engineering challenges, as every effort in working with these embeddings had to be multiplied by three. Therefore the team decided to investigate the possibility of learning universal set of embeddings. It turned out to be possible, and this reuse ended up simplifying their deployment pipelines as well as improving performance on individual tasks.
A broad selection of engineering problems that machine learning practitioners now face is given in Sculley et al. [52]. Most of them are considered anti-patterns in engineering, but are currently widespread in machine learning software. Some of these issues, such as abstraction boundaries erosion and correction cascades, are caused by the fact that ML is used in cases where the software has to take explicit dependency on external data. Others, such as glue code or pipeline jungles, stem from the general tendency in the field to develop general-purpose software packages. Yet another source of problems discussed in the paper is the configuration debt, which is caused by the fact that ML systems, besides all configurations a regular software systemmay require, add a sizable number of ML-specific configuration settings that have to be set and maintained.
Researchers and software engineers often find themselves working together on the same project aiming to reach a business goal with a machine learning approach. On surface there seems to be a clear separation of responsibilities: researchers produce the model while engineers build infrastructure to run it. In reality, their areas of concern often overlap when considering the development process, model inputs and outputs and performance metrics. Contributors in both roles often work on the same code. Thus it is beneficial to loop researchers into the whole development journey, making sure they own the product code base along with the engineers, use the same version control and participate in code reviews. Despite obvious onboarding and slow-start challenges, this approach was seen to bring

Files\\DLHub~ Simplifying publication, discovery, and use of machine learning models in science

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On
No	Google Scholar	0.0257	5			
				1	S	07/02/2022 22:52

There is a growing need for “learning systems” to support various phases in the ML lifecycle. While others have focused on supporting model development, training, and inference, few have focused on the unique challenges inherent in science, such as the need to publish and share models and to serve them on a range of available computing resources. In this paper, we present the

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On
				2	S	07/02/2022 22:54

However, scientific use of ML has specialized requirements, including the following. Publication, citation, and reuse: The scholarly process is built upon a
common workflow of publication, peer review, and citation. Progress is dependent on being able to locate, verify, and extend prior research, and careers are built upon publications and citation. As scholarly objects, ML models should be subject to similar publication, review, and citation models. Lacking standard methods for doing so, (a) many models associated with published literature are not available <23>; and (b) researchers adopt a range of ad hoc methods (from customized websites to GitHub)

				3	S	07/02/2022 22:54

Reproducibility: Concerns about reproducibility are having a profound
effect on research <27>. While reproducibility initiatives have primarily focused on making data and experimental processes available to reproduce findings, there is a growing interest in making computational methods available as well <28; 29; 30>.

				4	S	07/02/2022 22:54

Research infrastructure: While industry and research share common re-
quirements for scaling inference, the execution landscape differs. Researchers often want to use multiple (often heterogeneous) parallel and distributed computing resources to develop, optimize, train, and execute models. Examples include: laboratory computers, campus clusters, national cyberinfrastructure (e.g., XSEDE <31>, Open Science Grid <32>), supercomputers, and clouds. They often have their own resources that they would like to use for inference. Thus, learning systems need to support execution on different resources and enable migration between resources. Scalability: Large-scale parallel and distributed computing environments
enable ML models to be executed at unprecedented scale. Researchers require learning systems that simplify training and inference on enormous scientific datasets and that can be parallelized to exploit large computing resources. Low latency: ML is increasingly being used in real-time scientific pipelines,
for example to process and respond to events generated from sensor networks; classify and prioritize transient events from digital sky surveys for exploration; and to perform error detection on images obtained from X-ray light sources. There is a need in each case for low latency, near real-time ML inference for anomaly/error detection and for experiment steering purposes. As both the number of devices and data generation rates continue to grow, there is also a need to be able to execute many inference tasks in parallel, whether on centralized or "edge" computers. Research ecosystem: Researchers rely upon a large and growing ecosys-
tem of research-specific software

				5	S	07/02/2022 22:54

Model in the loop: Scientific analyses often involve multiple steps, such
as the staging of input data for pre-processing and normalization, extraction of pertinent features, execution of one or more ML models, application of uncertainty quantification methods, post-processing

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On

Nodes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\ML Model Engineering\Model traceability

PDF

Files\\AI lifecycle models need to be revised

No	Google Scholar	0.0178	1			
				1	S	24/02/2022 13:10

4.6 Model Evaluation
An essential step in the evaluation of the model is communicating how well the model performs according to the defined metrics. It is about demonstrating that the model meets business and regulatory needs and assessing the design of the model. One key difference between the metrics used in this step and the metrics used for Model Scoring is that these metrics are communicated to different stakeholders that do not necessarily have a Machine Learning or data science background. Thus, the set of metrics needs to be extended to a general audience. One complementary strategy used by practitioners is having live demos of the model with business stakeholders (P03, P15, P16). These demos allow stakeholders to try out different inputs and try corner cases.
4.6.1 Model Risk Assessment
An important aspect of evaluating a model at ING is making sure it complies with regulations, ethics, and organizational values (P15, P06). This is a common task for any type of model built within the organization – i.e., not only Machine Learning models but also economic models, statistical forecasting models, and so on. In the interviews, Model Risk Assessment was mentioned as mandatory within the model governance strategy, undertaken in collaboration with an independent specialized team (P06, P14). This is a long-stablished stage which is now being challenged by the specifics of Machine Learning. For example, traditional risk assessment teams did not initially have the right Machine Learning expertise to evaluate the models with confidence. Depending on the criticality level of the model, the intensity of the review may vary.
Each model owner is responsible for the risk management of their model, but colleagues from the risk department help and challenge the model owner in this process. During the periodic risk assessment process, assessors inspect the documentation pro-
vided by the Machine Learning team to assess whether all regulations and minimum standards are followed. The documentation used in this stage is considered to be overly time-consuming, as emphasized by P07: “70% percent of the time people are writing Word documents to explain their code is compliant.”. Although the process is still under development within ING, the following key points are being covered (P06): 1) model identification
95 Page 16 of 29 Empir Software Eng (2021) 26: 95
(identify if the candidate is a model which needs risk management), 2) model boundaries (define which components are part of the model), 3) model categorization (categorize the model into the group of models with a comparable nature, e.g. anti-money-laundering), 4) model classification (classify the model into in the class of models which require a comparable level of model risk management), and 5) assess the model by a number of sources of risk.

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On

Files\\Challenges in Deploying Machine Learning~ a Survey of Case Studies

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On
No	Google Scholar	0.0311	1			
				1	S	24/02/2022 11:03

5 Model Verification
The goal of the model verification stage is multifaceted, because an ML model should generalize well to unseen inputs, demonstrate reasonable handling of edge cases and overall robustness, as well as satisfy all functional requirements. In this section, we discuss issues concerning three steps within model verification: requirement encoding, formal verification and test-based verification.
7
5.1 Requirement encoding
Defining requirements for a machine learning model is a crucial prerequisite of testing activities. It often turns out that an increase in model performance does not translate into a gain in business value, as Booking.com discovered after deploying 150 models into production [44]. Therefore more specific metrics need to be defined and measured, such as KPIs and other business driven measures. In the case of Booking.comsuch metrics included conversion, customer service tickets or cancellations. Cross-disciplinary effort is needed to even define such metrics, as understanding frommodeling, engineering and business angles is required. Once defined, these metrics are used for monitoring of the production environment and for quality control of model updates.
Besides, simply measuring the accuracy of the ML model is not enough to understand its performance. Essentially, performance metrics should reflect audience priorities. For instance Sato et al. [45] recommend validating models for bias and fairness, while in the case described by Wagstaff et al. [31] controlling for consumption of spacecraft resources is crucial.
5.2 Formal Verification
The formal verification step verifies that the model functionality follows the requirements defined within the scope of the project. Such verification could include mathematical proofs of correctness or numerical estimates of output error bounds, but as Ashmore et. al. [14] point out this rarely happens in practice. More often quality standards are being formally set via extensive regulatory frameworks.
An example of where ML solutions have to adhere to regulations is the banking industry [46]. This requirement was developed in the aftermath of the global financial crisis, as the industry realized that there was a need for heightened scrutiny towards models. As a consequence an increased level of regulatory control is now being applied to the processes that define how the models are built, approved and maintained. For instance, official guidelines has been published by the UK's Prudential Regulation Authority [47] and European Central Bank [48]. These guidelines require model risk frameworks to be in place for all business decision-making solutions, and implementation of such frameworks requires developers to have extensive tests suites in order to understand behavior of their MLmodels. The formal verification step in that context means ensuring that the model meets all criteria set by the corresponding regulations.
Regulatory frameworks share similarities with country-wide policies, which we discuss in greater details in Section 7.1.
5.3 Test-based Verification
Test-based verification is intended for ensuring that the model generalizes well to the previously unseen data. While collecting validation dataset is usually not a problem, as it can be derived from splitting the training dataset, it may not be enough for production deployment.
In an ideal scenario testing is done in a real-life setting, where business driven metrics can be observed, as we discussed in Section 5.1.
Full scale testing in real-world environment can be challenging for a variety of safety, security and scale reasons, and is often substituted with testing in simulation. That is the case for models for autonomous vehicles control [26]. Simulations are cheaper, faster to run, and provide flexibility to create situations rarely encountered in real life. Thanks to these advantages, simulations are becoming prevalent in this field. However, it is important to remember that simulation-based testing hinges on assumptions made by simulation developers, and therefore cannot be considered a full replacement for real-world testing. Even small variations between simulation and real world can have drastic effects on the system behavior, and therefore the authors conclude that validation of the model and simulation environment alone is not enough for autonomous vehicles. This point is emphasized further by the experiences from the field of reinforcement learning [25], where use of simulations is a de-facto standard for training agents.
In addition, the dataset itself also needs to be constantly validated to ensure data errors do not creep into the pipeline and do not affect the overall quality. Breck et al. [49] argue that one of the most common scenarios when issues in data can go unnoticed is the setup where data generation is decoupled from the ML pipeline. There could be multiple reasons for such issues to appear, including bugs in code, feedback loops, changes in data dependencies. Data errors can propagate

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On

Files\\MSR4ML~ Reconstructing Artifact Traceability in Machine Learning Repositories

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On
No	Web of science	0.0140	1			
				1	S	07/02/2022 11:04

The increasing popularity of Machine Learning
(ML) is generating challenges also for developers. The multitude of programming languages, libraries and available resources allow them to easily build their own models or algorithms. However, ML models are tightly connected to their data implying a different development process from other types of software. Software projects often rely on version control platforms, such as GitHub, but these platforms have not yet been extended to support ML projects. There is poor support for data versioning and no link between ML and software artifacts. Thus, traceability and model evolution can become challenging for developers. While some specific ML platforms exist, they still require considerable manual specification of ML artifacts and links between them.

Nodes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\ML Model Engineering\Model training

PDF

Files\\500+ times faster than deep learning~ a case study exploring faster methods for text mining stackoverflow

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On
No	ACM Digital library	0.0081	1			
				1	S	11/02/2022 13:33

Deep learning methods are useful for high-dimensional data and are becoming widely used in many areas of software engineering. Deep learners utilizes extensive computational power and can take a long time to train– making it difficult to widely validate and repeat and improve their results. Further, they are not the best solution in all domains. For example, recent results show that for finding related Stack Overflow posts, a tuned SVM performs similarly to a deep learner, but is significantly faster to train. This paper extends that recent result by clustering the dataset,
then tuning every learners within each cluster. This approach is over 500 times faster than deep learning (and over 900 times faster ifwe use all the cores on a standard laptop computer). Significantly, this faster approach generates classifiers nearly as good (within 2% F1 Score) as the much slower deep learning

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On

Files\\AI lifecycle models need to be revised

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On
No	Google Scholar	0.0234	1			
				1	S	24/02/2022 13:12

4.4 Modeling
Model training is mostly done in on-premises environments such as Hadoop2 and Spark3 clusters (P09) or in generic systems using, for example, the scikit-learn4 library (P01). These private platforms are connected with the data lakes where data is stored, so training can be done on (a copy of) real production data (P01, P03). The on-premises environment has no outgoing connection to the internet, so a connection to other cloud services such as Microsoft Azure5 or Google Cloud6 is not possible (P08). This means that data scientists are limited to the tools and platforms available within the organization when dealing with sensitive data. Also, all project dependencies need to be previously approved, after which they are made available in a private package repository (P04, P12), which contains whitelisted packages that have been internally audited. This can be frustrating, when new ground-breaking AI technologies appear, practitioners have to wait before they can explore the potential of those technologies at ING (P12) – we later refer to this challenge as Technology Access (cf. Section 5). Fewer restrictions are in place if Machine Learning is applied to public data, for example on stock prices. In that case, external cloud services and packages may be used (P09).
2Hadoop enables distributed processing of large data sets across clusters of computers https://hadoop.apache. org 3Spark is a unified analytics engine for large-scale data processing. https://spark.apache.org 4Scikit-learn is a Machine Learning library for Python. https://scikit-learn.org 5Microsoft Azure is a cloud computing service. https://azure.microsoft.com/en-us 6Google Cloud is a cloud computing service. https://cloud.google.com
95 Page 14 of 29 Empir Software Eng (2021) 26: 95 Model training is an iterative process. Usually, multiple models are created for the same problem. First, a simple model is created (e.g., a linear regression model) to set as a baseline (P09). In the following iterations, more advanced models are compared to this baseline model. If an approach other than Machine Learning already exists (e.g., rule-based software), the models are also compared with this. To keep track of different versions of models, different teams use different strategies. For example, the team of P08 keeps track of an experiment log using a spreadsheet, in which the training set, validation set, model, and pre-processing steps are specified for each version. This approach for versioning is preferred over solutions like MLFlow7 for the sake of simplicity (P08, P15).
4.4.1 Model Scoring
An implicit sub stage of modeling is assessing model performance to measure how well the predictions of the model represent ground truth data. We define Model Scoring as assessing the performance of the model based on scor-
ing metrics (e.g., f1-score for supervised learning). It is also known as Validation by the Machine Learning community, which should not be confused with the definition by the Software Engineering community8 (Ryan and Wheatcraft 2017; 15288 2015). The main remarks for this stage are related to defining the right set of metrics (P03, P06,
P12, P14, P15, P16). The problem is two-fold: 1) identify the right metrics and 2) communicate why the selected metrics are right. Practitioners report that this is very problem-specific. Thus, it requires a good understanding of the business, data, and learning algorithms being used. From an organization's point of view, these different perspectives are a big barrier to defining validation standards. Thechallenges in Modeling summarize as follows: 1) thelatest MachineLearning
technologies are not alwayseligible for use;2)baseline models are essential artifacts for model development; 3) teams keep track of all experiments, which often revolves around keeping a customized spreadsheet; and4)defining performance metrics is problem-specific,

Files\\All versus one~ an empirical comparison on retrained and incremental machine learning for modeling performance of adaptable software

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On
No	Scopus	0.0508	9			
				1	S	10/02/2022 11:25

Given the ever-increasing complexity of adaptable
software systems and their commonly hidden internal information (e.g., software runs in the public cloud), machine learning based performance modeling has gained momentum for evaluating, understanding and predicting software performance, which facilitates better informed self-adaptations. As performance data accumulates during the run of the software, updating the performance models becomes necessary. To this end, there are two conventional modeling methods: the retrained modeling that always discard the old model and retrain a new one using all available data; or the incremental modeling that retains the existing model and tunes it using one newly arrival data sample. Generally, literature on machine learning based performance modeling for adaptable software chooses either of those methods according to a general belief, but they provide insufficient evidences or references to justify their choice.

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On
				2	S	10/02/2022 11:39

One fundamental to effective application of machine learn-
ing in performance modeling is the data, which determines the levels of knowledge that a model can learn and generalize. However, many real world scenarios do not have sufficient data, or the available data do not adequately represent what the adaptable software is likely to behave in changing and uncertain environments. Therefore, modeling software performance at runtime with evolving data stream has been increasingly important [8] [9]. Machine learning based performance modeling at runtime has the advantage that the model can be updated using the most up-to-date data samples, which inherently improves the effectiveness of the model.

				3	S	10/02/2022 11:39

For modeling performance at runtime, the problem that a
software engineer would face is: how to update the model when using a learning algorithm2 under evolving data? Literature from the Software Engineering and Machine Learning communities take two predominate modeling methods to achieve this: (i) either completely retraining the model by learning a new data sample in conjunction with the historical ones (i.e., the retrained modeling), or (ii) simply tuning the existing model using a new data sample as it arrives (i.e., the incremental modeling). The choice between those two methods does not change the interpretation of the model, but they make fundamentally different assumptions about how a model is learned and hence they lead to different variants of a learning

				4	S	10/02/2022 11:42

The Retrained and Incremental Modeling Modeling the performance of adaptable software via ma-
chine learning often require the model to learn whenever newly observed data sample becomes available as the software runs. However, the problem that a software engineer would face is: how to update the model when using machine learning under evolving data? According to the literature from both the Software Engineering and the Machine Learning community, there are two predominate modeling methods to achieve this: Retrained modeling: retrained modeling is similar to the traditional offline learning, where the old model is discarded and a new model is retrained using whatever data that is available, i.e., the new data samples and all the historical ones. The good side of retrained modeling is that it is able to capture the interrelation between different data samples given the fact that they are always learned in conjunction with each others.

				5	S	10/02/2022 11:42

Incremental modeling: incremental modeling follows the
online learning paradigm, which is truly incremental in the sense that instead of replacing the entire model, its internal structure is tuned using the new data sample. In other words, it learns each new data sample in isolation as they arrive. The good side of incremental modeling is the likely small computation effort. However, the fact that each data sample is learned individually may ignore some joint correlations that can only be discovered when data samples are learned in conjunction with each others, which may affect the accuracy.

				6	S	10/02/2022 11:43

Prior Retrained Performance Modeling To build machine learning based performance models under
evolving data stream, a large amount of research has relied on retrained modeling. Among others, Kundu et al. [15][16] have relied on Multi-Layer Perceptron (MLP) [10] and Support Vector Machine (SVM) [25] to model the performance of cloudbased and service-oriented software. Their models are built in the retrained manner, where certain amount of historical data is used to train the MLP model at design time, then at runtime, such a model is retrained whenever new data sample is available. Similarly, Siegmund et al. [20], Sieber et al. [17] and Gerostathopoulos et al. [26] use Linear Regression (LR) [27] to build the performance model at runtime, but again, the model is retrained completely instead of being tuned when significant outliers are detected or as new data is collected. Another notable effort of retrained modeling based on the Decision Tree (DT) family (e.g., M5 decision tree [28]), such as FUSION [18] and Guo et al. [19], where the performance model is discarded and rebuilt using all the available data when the adaptable software collects new information.

				7	S	10/02/2022 11:43

Prior Incremental Performance Modeling The other direction of effort on performance modeling
assumes truly incremental modeling. For example, incremental modeling has been used in relatively simpler learning algorithms, e.g., linear regression (e.g., in [12][14]) and ARMA (e.g., in [13]), when modeling performance under changing environment of an adaptable software. The linear nature of those models make incremental modeling much more straightforward and can be tuned using Recursive

				8	S	10/02/2022 11:44

The Comparison Procedure and Metrics To ensure generality, we investigated a wide range of combinations on scenarios and cases, which are defined as:
— Scenario: A scenario refers to each pair of learning algorithm and performance indicator of a software, e.g., using LR to predict the throughput of ASOS.

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On
				9	S	10/02/2022 11:45

Accuracy (Error): We measure the accuracy of the model as
the adaptable software runs and as the model evolves4. At each time point t, a model is firstly updated by the data samples up to t-1 (t-2 for environment features). Then in the validation phase, the model takes the adaptable features at t and the environment features at t-1 to predict the performance at t, which is then compared with the ground truth at t. Given a scenario, we adopt Mean Absolute Error (MAE) to show the accuracy over all the intervals and repeated runs of a case, as it can additionally reflect the practicality of the error in the original scale. Suppose yk,t and ˆyk,t are the predicted and actual performance of the kth run at time t respectively; the MAE over n intervals and m repeated runs is:
MAE = 1 × m× n
~m k=1
~n t=1
|yk,t − ˆyk,t| (3) Training Time: We collected the time taken for training,
and analyzed the Mean Training Time (MTT) over all the time intervals and repeated runs of a case. Robustness: By analyzing the variance of the accuracy
and training time, we aim to understand the robustness of

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On

Files\\Challenges in Deploying Machine Learning~ a Survey of Case Studies

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On
No	Google Scholar	0.0416	1			
				1	S	24/02/2022 11:03

4 Model Learning
Model learning is the stage of the deployment workflow that enjoys the most attention within the academic community. All modern research in machine learning methods contributes towards better selection and variety ofmodels and approaches that can be employed at this stage. As an illustration of the scale of the field's growth, the number of submissions to NeurIPS, primary conference on ML methods, has quadrupled in six years, going from 1678 submissions in 2014 to 6743 in 2019 [29]. Nevertheless, there is still plenty of practical considerations that affect the model learning stage. In this section, we discuss issues concerning three steps within model learning: model selection, training and hyper-parameter selection.
4.1 Model selection
In many practical cases the selection of a model is often decided by one key characteristic of a model: complexity. Despite areas such as deep learning and reinforcement learning gaining increasing levels of popularitywith the research community, in practice simpler models are often chosen as we explain below. Such model include shallow network architectures, simple PCA-base approaches, decision trees and random forests.
Simple models can be used as a way to prove the concept of the proposed ML solution and get the end-to-end setup in place. This approach accelerates the time to get a deployed solution, allows the collection of important feedback and also helps avoid overcomplicated designs. This was the case reported by Haldar et al. [30]. In the process of applying machine learning to AirBnB search, the team started with a complex deep learning model. The team was quickly overwhelmed by its complexity and ended up consuming development cycles. After several failed deployment attempts the neural network architecture was drastically simplified: a single hidden layer NN with 32 fully connected ReLU activations. Even such a simple model had value, as it allowed the building of a whole pipeline of deploying ML models in production setting, while providing reasonably good performance2. Over time the model evolved, with a second hidden layer being added, but it still remained fairly simple, never reaching the initially intended level of complexity.
Another advantage that less complex models can offer is their relatively modest hardware requirements. This becomes a key decision point in resource constrained environments, as shown by Wagstaff et al. [31]. They worked on deploying ML models to a range of scientific instruments onboard Europa Clipper spacecraft. Spacecraft design is always a trade-off between the total weight, robustness and the number of scientific tools onboard. Therefore computational resources are scarce and their usage has to be as small as possible. These requirements naturally favor the models that are light on computational demands. The team behind Europa Clipper used machine learning for three anomaly detection tasks, some models took time series data as input and some models took images, and on all three occasions simple threshold or PCA based techniques were implemented. They were specifically chosen because of their robust performance and low demand on computational power.
A further example of a resource-constrained environment is wireless cellular networks, where energy, memory consumption and data transmission are very limited. Most advanced techniques, such as deep learning, are not considered yet for practical deployment, despite being able to handle highly dimensional mobile network data [32].
The ability to interpret the output of a model into understandable business domain terms often plays a critical role in model selection, and can even outweigh performance considerations. For that reason decision trees (DT), which can be considered a fairly basic ML algorithm, are widely used in practice. For example, Hansson et al. [33] describe several cases in manufacturing that adopt DT due to its high interpretability.
Banking is yet another example of an industry where DT finds extensive use. As an illustrative example, it is used by Keramati et al. [34] where the primary goal of the ML application is to predict customer churn by understanding if-then rules. While it is easy to imagine more complicated
2We discuss more benefits of setting up the automated deployment pipeline in Section 6.3. 6
models learning the eventual input-output relationship for this specific problem, interpretability is key requirement here because of the need to identify the features of churners. The authors found DT to be the best model to fulfill this requirement.
Nevertheless, deep learning (DL) is commonly used for practical background tasks that require analysis a large amount of previously acquired data. This notion is exemplified by the field of unmanned aerial vehicles (UAV) [35]. Image sensors are commonplace in UAVs due to their low cost, low weight, and low power consumption. Consequently, processing images acquired from sensors is the main way of exploiting excellent capabilities in processing and presentation of raw data that DL offers. But computational resource demands still remain the main blocker for deploying DL as an online processing instrument on board of UAVs.
4.2 Training
One of the biggest concern with model training is the economic cost associated with carrying the training stage due to the computational resources required. This is certainly true in the field of natural language processing (NLP), as illustrated by Sharir et al. [36]. The authors observe that while the cost of individual floating-point operations is decreasing, the overall cost of training NLP is only growing. They took one of the state-of-the-art models in the field, BERT [37], and found out that depending on the chosen model size full training procedure can cost anywhere between $50k and $1.6m, which is unaffordable for most research institutions and even companies. The authors observe that training dataset size, number ofmodel parameters and number of operations utilized by the training procedure are all contributing towards the overall cost. Of particular importance here is the second factor: novel NLP models are already using billions of parameters, and this number is expected to increase further in the nearest future [38].
A related concern is raised by Strubell et al. [39] regarding the impact the training ofML models has on the environment. By consuming

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On

Nodes\\Maintainable ML\\Challenges in Maintaining a ML systems and applications\ML Model Engineering\Testing

PDF

Files\\Automatic Unit Test Generation for Machine Learning Libraries~ How Far Are We~

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On
No	ACM Digital library	0.0112	3			
				1	S	08/02/2022 13:30

Automatic unit test generation that explores the in-
put space and produces effective test cases for given programs have been studied for decades. Many unit test generation tools that can help generate unit test cases with high structural coverage over a program have been examined. However, the fact that existing test generation tools are mainly evaluated on general software programs calls into question about its practical effectiveness and usefulness for machine learning libraries, which are statistically-orientated and have fundamentally different nature and construction from general

				2	S	08/02/2022 13:31

We are witnessing a wide adoption of Machine Learning
(ML) models in many software systems lately. Software applications powered by ML are being used in critical sectors of our daily lives; from finance and energy, to health and transportation [9, 10, 11]. Thus, building reliable and secure ML systems has become an increasingly critical challenge for software developers. However, ML libraries are often statistically-orientated, and have fundamentally different nature and construction compared to general software projects [10, 12], which makes the usefulness of existing automatic test generation tools on them unknown.

				3	S	08/02/2022 13:34

Current unit test suite in ML libraries has lower quality regarding code coverage (on average, 34.1%) and mutation score (on average, 21.3%). In addition, the testing effort of academic-led ML libraries is unbalanced distributed and their unit test quality is significantly worse than that of community-led ML libraries.

Files\\Automatically Authoring Regression Tests for Machine-Learning Based Systems

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On
No	Web of science	0.0277	5			
				1	S	08/02/2022 13:21

Two key design characteristics of machine learning
(ML) systems—their ever-improving nature, and learning-based emergent functional behavior—create a moving target, posing new challenges for authoring/maintaining functional regression tests.

				2	S	08/02/2022 13:23

End-to-end regression testing of Machine Learning (ML)
software has disrupted the way we think about functional testing [1], [2]. Traditional functional tests are of the form (input, expected output, assertion()), where input is supplied to the software under test (SUT), and the test oracle (expected output and assertion()) verifies whether the SUT functioned as expected [3]. Testers strive to develop a test suite that provides adequate coverage of software features [4]. Regression testing of ML systems casts aside the 3 tra-
ditional tenets of functional testing: input, expected output, and coverage in multiple ways. First, the input spaces of ML-based systems are extremely large [5] (think about all the situations to which an autonomous vehicle must react), which is why these systems are, by design, optimized for their most common inputs. Indeed, they may not always return correct outputs for all uncommon inputs. Developers may not even know all the uncommon/corner cases [6] at design time, neither would the testers during in-house test development [7]. The software's eventual functional behavior is not pre-defined; rather it emerges as it learns and evolves. Second, imperfect understanding of the input space upsets

	Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On
					3	S	08/02/2022 13:24

the traditional role of functional testing, which is to use functional boundaries/partitions and corner cases to ensure that the system behaves as intended within—and at the boundaries of—each partition. Consequently, test authors are unable to determine whether they have an adequate test suite that covers all functional boundaries. All their hard-coded inputs in a test suite may be distributed over an initial guesstimated set of partitions but may, over time, end up in quite another set, causing the inputs to become less important or even irrelavent. Moreover, because much of the ML decision logic is typically encoded mathematically, e.g., in a deep neural network or a logistic regression model, there is no control-flow-graph, and hence traditional coverage also does not directly apply [8]

					4	S	08/02/2022 13:24

Third, ML systems, by their very nature, are designed
to better serve the most prominent inputs and constantly improve their outputs over time by learning from new training data [7]. A traditional test oracle will quickly become obsolete as its hard-coded expected output/assertion() turns stale with respect to the software's new improved output

					5	S	08/02/2022 13:24

Finally, another distinction in regression testing of ML vs.
conventional systems is that individual test failures for ML systems may not be indicative of a bug. Recall that ML systems are optimized for certain classes of common inputs – they may not work for uncommon inputs; hence failures on such inputs may be perfectly acceptable. Instead, of interest to the ML-system developer are systematic test failures as well as patterns of failures that assist in software/model debugging. This shift creates new challenges for test authors, who must now create a large number of tests to reveal such patterns. Moreover, test failure triage is not always useful when looking at individual isolated failures; rather, groups of failing tests need to be examined to provide a more holistic picture of what went wrong with the ML software.

Files\\Cats are not fish~ deep learning testing calls for out-of-distribution awareness

	Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On
	No	ACM Digital library	0.0409	6			
					1	S	08/02/2022 12:35

As Deep Learning (DL) is continuously adopted in many industrial applications, its quality and reliability start to raise concerns. Similar to the traditional software development process, testing the DL software to uncover its defects at an early stage is an effective way to reduce risks after deployment. According to the fundamental assumption of deep learning, the DL software does not provide statistical guarantee and has limited capability in handling data that falls outside of its learned distribution, i.e., out-of-distribution (OOD) data.

					2	S	08/02/2022 12:38

However, different from traditional software whose decision
logic is mostly programmed by the developer, deep learning adopts a data-driven programming paradigm. In particular, the major tasks of a DL developer are preparing the training data, labeling the data, programming the architecture of the deep neural network (DNN), and specifying the training configuration. All the decision logic is automatically learned during the runtime training phase and encoded in the obtained DNN (e.g., by weights, bias, and their combinations). Due to the differences of programming paradigm, the logic encoding format, and the tasks that a DNN is often developed for (e.g., image recognition), testing techniques for traditional software cannot be directly applied and new testing techniques are needed for DNNs. While some recent progress has been made in proposing novel testing criteria [17, 25, 33, 35] and test generation techniques for quality assurance of DNNs [8, 33, 35, 43, 48, 55, 58], it still lacks interpretation and understanding on the detected errors by such techniques and their impact. For example, it is not clear whether errors are indeed caused by missing training data or insufficient training, etc. The fundamental assumption of deep learning is that

					3	S	08/02/2022 12:38

If the new unseen input data has a similar distribution as the
training data, deep learning provides some statistical guarantee on its prediction correctness in terms of accuracy. However, if the new input data does not follow the training data (i.e., out-of-distribution (OOD)), deep learning does not provide statistical guarantee on its prediction. For example, if a DNN is only trained on cat and dog data for binary classification, given an input data offi sh, the DNN can still produce a prediction result. However, this input data does not follow the distribution of cat and dog data. Hence, handling the fish data goes beyond the capability of this DNN and should not be considered as valid input. Intuitively, erroneous inputs that follow the distribution of train-
ing data may reveal the real weakness of the DNN since the DNN is expected to handle such data. On the other hand, input errors that are considered out-of-distribution may either inherit new information benefitting generalization as well as a domain shift or are simply irrelevant to the DL application. Thereby, the root cause of an error may be identified through analyzing its distribution behavior, which

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On
				4	S	08/02/2022 12:40

To summarize, this paper makes the following contributions:
• We perform a large scale empirical study on how deep learning testing affects the data distribution of the generated test cases; and how distribution aware testing influences DNN model robustness.
• Our study identifies the impact of mutation operators and coverage criteria on the distribution of the generated test cases. We find that image rotation, contrast and brightness tend to generate more ID data while image blur is more likely to generate OOD data. In terms of the coverage criteria, NBC and SNAC facilitate to generate more OOD data than others.
• We demonstrate the effectiveness of distribution aware retrain-
ing, outperforming the state-of-the-art by up to 21.5%. Based on our results, we provide guidelines on distribution-aware error selection for robustness enhancement, by studying the effect of root cause of ID and OOD errors.

				5	S	08/02/2022 12:40
				6	S	08/02/2022 12:41

• AUROC. Given an unknown input, OOD detection techniques need to identify a threshold to classify it as ID or OOD. The area under the receiver operating characteristic curve (AUROC) [14] is usually used to evaluate the performance of a classification method across multiple thresholds. The AUROC can be thought of as the probability that an anomalous example is given a higher OOD score than an in-distribution example [16]. Thus, the higher AUROC, the better the OOD detector.
• TPRN, which is the true positive rate at N% true negative rate (TPRN). We regard OOD data as the positive class. First, we use N% true negative rate to select one threshold for the OOD detector. Then, with this threshold, we evaluate the true positive rate of the detector. Note that, for the parameter N in TPRN, a larger N means we select a bigger threshold such that more data is perceived under the threshold as ID (i.e., higher true negative rate). Thus, a larger N provides more confident measurement for detecting OOD data while a smallerN provides more confident measurement for detecting ID data.

Files\\Machine Learning Testing~ Survey, Landscapes and Horizons

No	Google Scholar	0.0240	26			
				1	S	07/02/2022 12:59

Safety-critical applications such as self-driving systems [1], [2] and medical treatments [3], increase the importance of behaviour relating to correctness, robustness, privacy, efficiency and fairness. Software testing refers to any activity that aims to detect the differences between existing and required behaviour [4]. With the recent rapid rise in interest and activity, testing has been demonstrated to be an effective way to expose problems and potentially facilitate to improve the trustworthiness of machine learning systems.

				2	S	07/02/2022 13:00

Machine learning testing poses challenges that arise from
the fundamentally different nature and construction of machine learning systems, compared to traditional (relatively more deterministic and less statistically-orientated) software systems. For instance, a machine learning system inherently follows a data-driven programming paradigm, where the decision logic is obtained via a training procedure from training data under the machine learning algorithm's architecture [8]. The model's behaviour may evolve over time, in response to the frequent provision of new data [8]. While this is also true of traditional software systems, the core underlying behaviour of a traditional system does not typically change in response to new data, in the way that a machine learning system can. Testing machine learning also suffers from a particularly pernicious instance of the Oracle Problem [9]. Machine learning systems are difficult to test because they are designed to provide an answer to a question for which no previous answer exists [10]. As Davis and Weyuker said [11], for these kinds of systems 'There would be no need to write such programs, if the correct answer were known'. Much of the literature on testing machine learning systems seeks to find techniques that can tackle the Oracle problem, often drawing on traditional software testing approaches.

				3	S	07/02/2022 13:00

The behaviours of interest for machine learning systems
are also typified by emergent properties, the effects of which can only be fully understood by considering the machine learning system as a whole. This makes testing harder, because it is less obvious how to break the system into smaller components that can be tested, as units, in isolation. From a testing point of view, this emergent behaviour has a tendency to migrate testing challenges from the unit level to the integration and system level. For example, low accuracy/ precision of a machine learning model is typically a composite effect, arising from a combination of the behaviours of different components such as the training data, the learning program, and even the learning framework/library [8]. Errors may propagate to become amplified or suppressed, inhibiting the tester's ability to decide where

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On
				4	S	07/02/2022 13:01

In this paper, we use the term ‘Machine Learning Testing’
(ML testing) to refer to any activity aimed at detecting differences between existing and required behaviours of machine learning systems. ML testing is different from testing approaches that use machine learning or those that are guided by machine learning, which should be referred to as ‘machine learning-based testing’. This nomenclature accords with previous usages in the software engineering literature. For example, the literature uses the terms ‘state-based testing’ [16] and ‘search-based testing’ [17], [18] to refer to testing techniques that make use of concepts of state and search space, whereas we use the terms ‘GUI testing’ [19] and ‘unit testing’ [20] to refer to test techniques that tackle challenges of testing Graphical User Interfaces (GUIs) and code units.

				5	S	07/02/2022 13:01

				6	S	07/02/2022 13:02

Role ofTesting in ML Development Fig. 4 shows the life cycle of deploying a machine learning system with ML testing activities involved. At the very
beginning, a prototype model is generated based on historical data; before deploying the model online, one needs to conduct offline testing, such as cross-validation, to make sure that the model meets the required conditions. After deployment, the model makes predictions, yielding new data that can be analysed via online testing to evaluate how the model interacts with user behaviours. There are several reasons that make online testing essen-
tial. First, offline testing usually relies on test data, while test data usually fails to fully represent future data [42]; Second, offline testing is not able to test some circumstances that may be problematic in real applied scenarios, such as data loss and call delays. In addition, offline testing has no access to some business metrics such as open rate, reading time, and click-through rate.

				7	S	07/02/2022 13:03

Offline Testing The workflow of offline testing is shown by the top dotted rectangle of Fig. 5. At the very beginning, developers need to conduct requirement analysis to define the expectations of the users for the machine learning system under test. In requirement analysis, specifications of a machine learning system are analysed and the whole testing procedure is planned.

				8	S	07/02/2022 13:02

				9	S	07/02/2022 13:03

3.2.3 Online Testing Offline testing tests the model with historical data without in the real application environment. It also lacks the data collection process of user behaviours. Online testing complements the shortage of offline testing, and aims to detect bugs after the model is deployed online.

				10	S	07/02/2022 13:03

ML Testing Properties
Testing properties refer to what to test in ML testing: for what conditions ML testing needs to guarantee for a trained model. This section lists some typical properties that the literature has considered. We classified them into basic functional requirements (i.e., correctness and model relevance) and non-functional requirements (i.e., efficiency, robustness,3 fairness, interpretability). These properties are not strictly independent of each
other when considering the root causes, yet they are different external manifestations of the behaviours of an ML system and deserve being treated independently in ML testing.

				11	S	07/02/2022 13:04

This section organises ML testing research based on the test-
ing workflow as shown by Fig. 5. ML testing includes offline testing and online testing. Albarghouthi and Vinitsky [75] developed a fairness specification language that can be used for the development of runtime monitoring, in detecting fairness issues. Such a kind of run-time monitoring belongs to the area of online testing. Nevertheless, current research mainly centres on offline testing as introduced below. The procedures that are not covered based on our paper collection, such as requirement analysis and regression testing and those belonging to online testing are discussed as research opportunities in Section 10.

				12	S	07/02/2022 13:05

5.1 Test Input Generation

				13	S	07/02/2022 13:04

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On
				14	S	07/02/2022 13:05
5.2 Test Oracle						
				15	S	07/02/2022 13:05
5.3 Test Adequacy						
				16	S	07/02/2022 13:05
5.4 Test Prioritisation and Reduction						
				17	S	07/02/2022 13:05
5.5 Bug Report Analysis						
				18	S	07/02/2022 13:05
5.6 Debug and Repair						
				19	S	07/02/2022 13:05
5.7 General Testing Framework and Tools						
				20	S	07/02/2022 13:05
6ML PROPERTIES TO BE TESTED						
				21	S	07/02/2022 13:06
6.3 Robustness and Security						
				22	S	07/02/2022 13:06
6.4 Efficiency						
				23	S	07/02/2022 13:06
6.5 Fairness						
				24	S	07/02/2022 13:06
6.6 Interpretability						
				25	S	07/02/2022 13:06
6.7 Privacy						
				26	S	07/02/2022 13:08
Challenges in ML Testing As this survey reveals, ML testing has experienced rapid recent growth. Nevertheless, ML testing remains at an early stage in its development, with many challenges and open questions lying ahead. Challenges in Test Input Generation. Although a range of test input generation techniques have been proposed (see more in Section 5.1), test input generation remains challenging because of the						

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On

Files\\On testing machine learning programs

No	Web of science	0.0182	5			
				1	S	07/02/2022 10:17

Traditionally, software systems are constructed deductively, by writing down the rules that govern the behavior of the system as program code. However, with ML, these rules are inferred from training data (i.e.,, they are generated inductively). This paradigm shift in application development makes it difficult to reason about the behavior of software systems with ML components, resulting in systems that are intrinsically challenging to test and verify, given that they do not have (complete) specifications or even source code corresponding to some of their critical behaviors. In fact some ML programs rely on proprietary third-party libraries like

				2	S	07/02/2022 10:18

Conceptual issues. Once data is gathered, cleaning data tasks are required to ensure that the data is consistent, free from redundancy and given a reliable starting point for statistical learning. Common cleaning tasks include: (1) removing invalid or undefined values (i.e., Not-a-Number, Not-Available), duplicate rows, and outliers that seems to be too different from the mean value); and (2) unifying the variables' representations to avoid multiple data formats and mixed numerical scales. This can be done by data transformations such as normalization, min-max scaling, and data format conversion. This pre-processing step allows to ensure a high quality of raw data,

				3	S	07/02/2022 10:19

The identified patterns represent the core logic of the model. Changes in data (i.e., the input signals) are likely to have a direct impact on these patterns and hence on the behavior of the model and its corresponding predictions. Because of this strong dependence on data, ML models are considered to be data-sensitive or data-dependent algorithms. A poor selection of features can impact a ML system negatively. Sculley et al. [3] report that unnecessary dependencies to features that contribute with little or no value to the model quality can generate vulnerabilities and noises in a ML system. Examples of such features are : Epsilon Feature, which are features that have no significant contribution to the performance of the model, Legacy Feature, which are features that lost their added information value on model accuracy improvement when other more rich features are included in the model, or Bundled Features, which are groups of features that are integrated to a ML system simultaneously without a proper testing of the contribution of each individual feature.

				4	S	07/02/2022 10:19

Implementation issues. To process data as described above, ML engineers implement data pipelines containing components for data transformations, validation, enrichment, summarization, and–or any other necessary treatment. Each pipeline component is separated from the others, and takes in a defined input, and returns a defined output that will be served as input data to the next component in the pipeline. Data

				5	S	07/02/2022 10:20

Dead experimental code paths which happens when code is written for rapid prototyping to gain quick turnaround times by performing additional experiments simply by tweaks and experimental code paths within the main production code.

Files\\Software Framework for Data Fault Injection to Test Machine Learning Systems

No	Web of science	0.0308	2			
				1	S	03/02/2022 15:34

Data-intensive systems are sensitive to the quality
of data. Data often has problems due to faulty sensors or network problems, for instance. In this work, we develop a software framework to emulate faults in data and use it to study how machine learning (ML) systems work when the data has problems. We aim for flexibility: users can use predefined or their own dedicated fault models. Likewise, different kind of data (e.g. text, time series, video) can be used and the system under test can vary from a single ML model to a complicated software system. Our goal is to show how data faults can be emulated and how that can be used in the study and development of ML

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On
				2	S	03/02/2022 15:35

we face questions that not only influence the testing phase but also the development decisions. Such questions include the following: – Should we train the system with perfect or with faulty data? Examples of faulty data are far less common than examples of correct data but we may still have a good understanding of the kinds of data faults the system will face over its lifetime.
– Are some ML algorithms, architectures, or hyperparameter selections more robust towards data faults than others?
– How trustworthy the results of the algorithms are when used in real-life settings, which include faulty input data? The difficulty of making a system deal with data faults
comes from multiple sources. To begin with, data faults come in different forms. Some of them are systematic (e.g. sensor drift), whereas others are more random (e.g. a broken network connection). They happen infrequently so the training material there may not have many examples of faulty cases. Furthermore, it is not obvious what we should do to deal with faults – change the associated training data, change the model, or simply ignore the faulty output somehow. Unfortunately, testing how a system behaves with different kinds of data faults has been difficult

Files\\TensorFI~ A Configurable Fault Injector for TensorFlow Applications

Aggregate	Classification	Coverage	Number Of Coding Reference	Reference Number	Coded By Initials	Modified On
No	Scopus	0.0240	3			
				1	S	03/02/2022 15:05

TensorFlow is a high-level dataflow framework for building ML applications and has become the most popular one in the recent past. ML applications are also being increasingly used in safety-critical systems such as self-driving cars and home robotics. Therefore, there is a compelling need to evaluate the resilience of ML applications built using frameworks such as TensorFlow. In this paper, we build a high-level fault injection framework for TensorFlow called TensorFI for evaluating the resilience of ML applications. TensorFI is flexible, easy to use, and portable. It also allows ML application programmers to explore the effects of different parameters and algorithms on error

				2	S	03/02/2022 15:06

Fault Injection (FI) is a widely used technique to evaluate
the resilience of software applications to faults. While FI has been extensively used in general purpose applications, its use in ML applications presents three main challenges. First, because ML applications are often written using specialized infrastructures, it is difficult to inject faults at the level of individual program statements or variables as these are hidden inside the framework. Second, it is difficult to interpret the results of the FI experiments as they are dependent on the application and the inputs as well as the framework being deployed. Finally, performing FI in ML applications requires the programmer to understand where faults are likely to occur in the application and map them to its implementation.

				3	S	03/02/2022 15:08

Based on our results, we find that the error resilience of
ML applications can be very different under different algorithms and input datasets. Hence, ML applications need to be evaluated on a per application basis before their deployment, in order to benchmark their operational resilience. Further, we find that the error resilience (i.e., accuracy drops) of ML applications depends on the amount of output classes available in the input dataset used. This should be taken into consideration when designing resilient ML applications.