Out-of-the-box Reproducibility: A Survey of Machine Learning Platforms

Richard Isdahl
Department of Computer Science
Norwegian University of Science and Technology
Trondheim, Norway

Odd Erik Gundersen
Department of Computer Science
Norwegian University of Science and Technology
Trondheim, Norway
odderik@ntnu.no

***Abstract*—Even machine learning experiments that are fully conducted on computers are not necessarily reproducible. An increasing number of open source and commercial, closed source machine learning platforms are being developed that help address this problem. However, there is no standard for assessing and comparing which features are required to fully support reproducibility. We propose a quantitative method that alleviates this problem. Based on the proposed method we assess and compare the current state of the art machine learning platforms for how well they support making empirical results reproducible. Our results show that BEAT and Floydhub have the best support for reproducibility with Codalab and Kaggle as close contenders. The most commonly used machine learning platforms provided by the big tech companies have poor support for reproducibility.**

***Index Terms*—Reproducibility, reproducible AI, machine learning platforms, survey.**

I. Introduction

A concern has grown in the scientific community related to the reproducibility of scientific results. The concern is not unjustified. Baker reports that the scientific community is in agreement that there is a reproducibility crisis going on [1]. According to the findings of the ICLR 2018 Reproducibility Challenge, experts in machine learning have similar concerns about reproducibility; more worryingly, their concern increased after trying to reproduce research results [2]. In psychology, the reproducibility project was only able to reproduce 36 out of 100 psychology research articles with statistically significant results [3]. While reproducibility does not necessarily mean discovery of truth, as Devezer et al. [4] suggest, enabling reproducibility makes the analyses and evaluations transparent. Transparency enables errors to be found and corrected faster, ultimately shortening the path to uncovering the truth. Braun and Ong argue that computer science and machine learning should be in a better shape than other sciences, as many if not all experiments are completely conducted on computers [5].

Still, computer science and machine learning research is not necessarily reproducible. This was shown by Collberg and Proebsting who tried to execute code published as part of 601 papers. Their efforts succeeded in 32.1% of the experiments when not communicating with authors and 48.3% when communicating with the authors [6]. In their experiment, they only tried to run the code; they did not evaluate whether the results were reproducible. Does running the code mean that we can expect to reproduce the results? Henderson et al. investigated deep reinforcement learning and found that random variables, differing implementations of standard algorithms used for baseline comparison and performance evaluation, produced irreproducible results when they ran the same code on the same hardware platform [7]. Hong et al. showed that different hardware, compilers and compiler settings resulted in similar variance of the output as changing the initial conditions of weather simulations [8]. In other words, it is not possible to distinguish between simulations that have been run with the same initial conditions on different hardware, with different compiler settings and on different software platforms and simulations running on the exact same software and hardware with different initial conditions. Furthermore, Nagarajan et al. showed that it is possible to achieve deterministic results when computations are done on a graphics processing unit (GPU); the results will be completely different, but still deterministic, if run on a different GPU [9]. So, even if independent researchers are able to execute the code, the results are not guaranteed to be reproducible. There is also a lack of openness and version control of code and data [10], and the quality of documentation is poor [11].

Many solutions for solving the reproducibility issues in computer science and machine learning have been proposed, and some are mentioned here. As experiments are run on computers, it is possible to share the complete experiment as proposed by [12], [13]. Gil et al. suggested that the experiment procedures should be made explicit [14]. Sethi et al. even proposed auto-generating code for deep neural network architectures by analyzing research papers and in this way reproducing the results [15]. Executable notebooks, such as Jupyter Lab, have been proposed as solutions for reproducibility, but everyone does not agree that they are the silver bullet [16].

In addition to the suggested solutions, several recommendations for combating the reproducibility crisis have been made by Wilkinson et al. [17], Stodden et al. [18], Nosek et al. [3], Gil et al. [14], Starr et al. [19] and several others. The remedies are (i) openness and transparency in form of open sharing of code and data, but also open publishing, (ii) good documentation where the experiments, workflows and methods are described in detail, and (iii) version control of code, data and results, (iv) proper citation of code and data, (v) licenses so that it is clear how code and data can be used and finally (vi) preregistering of study designs to avoid p-hacking and HARKing.

The reproducibility crisis happens at the same time as there is an increased interest in machine learning and artificial intelligence. The top-conferences in AI and machine learning get increased attention. For example, AAAI[1] had 3800 submissions in 2018 and close to 8000 in 2019. For IJCAI[2], the numbers were 3500 in 2018 and 4500 in 2019. Lately, many machine learning platforms have been introduced, both open source and closed source, that run locally or in the cloud. Cloud solution providers such as Google, Amazon and Microsoft provide machine learning services as part of their cloud offering. Other solutions exist as well.

Goal: Given that (i) reproducibility is an issue, even in computer science and for machine learning, (ii) the increased interest in machine learning research and (iii) the amount of machine learning platforms that have been introduced lately, how easy is it to conduct research in machine learning that is well documented and reproducible? Our goal is to investigate how well current machine learning platforms support reproducibility out-of-the-box.

Contribution: Our contributions are threefold: (i) we propose a framework for comparing the support for reproducibility of machine learning frameworks, (ii) we conduct a survey of how well machine learning platforms support reproducibility and (iii) we analyse which features that should be developed for the different platforms in order to improve reproducibility support.

Results: The results of our survey show that no machine learning platform fully supports the feature set described by the proposed framework. More development is still needed for the surveyed platforms to fully support reproducibility.

The rest of this paper is organized as follows: Reproducibility is discussed in section 2. In section 3, a framework for quantifying reproducibility support is presented. The research method is presented in section 4. The survey platforms are introduced in section 5, while section 6 contains the results of the survey. The results are discussed in section 7. Section 8 concludes and provides some guidelines for potential future work.

II. Reproducibility

No ultimate definition of reproducibility is agreed upon. Instead, researchers have presented several competing definitions. Despite, the literature mostly agrees that reproducibility is not a boolean variable. An experiment is not reproducible or not reproducible; reproducibility comes in different shades. Drummond argues that replication means to exactly replicate the original experiment and that reproducibility is obtaining the same results from quite a different experiment [20]. Stodden states that replication is re-running the experiment with code and data provided by the author, while reproduction is a broader term that implies both replication and the regeneration of findings with at least some independence from the original code and/or data [21]. Peng suggests that reproducibility is a continuous variable ranging from only a paper describing an experiment being shared to the linked executable code and data being shared along with the paper [22]. Goodman et al. present three different terms describing reproducibility: 1) Methods reproducibility means that the exact same procedures could be exactly followed, 2) Results reproducibility refers to obtaining the same results from conducting an independent study whose procedures closely match the original study and 3) Inferential reproducibility in which qualitatively the same conclusions can be drawn from an independent study or reanalysis of the original study [23]. Gundersen and Kjensmo propose that for AI research three reproducibility degrees can be defined based on which documentation the original researchers share with independent researchers [11]. The documentation could be divided into (i) the scientific report, (ii) the data and (iii) the code from the original experiment. Tatman et al. suggest three levels also based on what is shared: 1) Low reproducibility: paper is shared, 2) Medium reproducibility: paper, code and data are shared, 3) High reproducibility: paper, code, data and environment is shared [24]. Plesser provides a good overview of different definitions [25].

III. Quantifying Support for Reproducibility

Our work build on the definition of reproducibility and the method for quantifying reproducibility that are suggested by Gundersen and Kjensmo [11]. They propose three factors, one for each documentation type, and specify variables that describe each of these three factors. The three factors are *Method*, describing the scientific report communicating methods and ideas to other researchers, *Data*, which is not only about sharing the data, but also indicating which parts were used for training, validation and testing, and *Experiment*, which is the code both for running the experiment and for any methods that are developed. Inspired by Gundersen et al. [26] we expand the set of variables from the 16 to 22. The idea is that the variables and factors are relevant for reproducing the results of empirical artificial intelligence research described in a scientific paper. The documentation quality and reproducibility degree of empirical AI research could be quantified by three metrics. We base our survey on the same idea, but instead of scoring a research paper on how reproducible it is, we assess how well machine learning platforms support reproducible empirical research by scoring the platforms on whether they have features that implement the variables. See table I for a description of the variables and the factors they belong to.

The three reproducibility metrics are defined as follows:

$$R1F(p) = \frac{\delta_1 Method(p) + \delta_2 Data(p) + \delta_3 Exp(p)}{\delta_1 + \delta_2 + \delta_3} \quad (1)$$

$$R2F(p) = \frac{\delta_1 Method(p) + \delta_2 Data(p)}{\delta_1 + \delta_2}, \quad (2)$$

[1]https://www.aaai.org
[2]https://www.ijcai.org

TABLE I
CONTAINS THE DEFINITIONS OF WHAT THE DIFFERENT VARIABLES MEAN, AND WHICH FACTORS THEY BELONG TO.

Factor	Variable	Description
Experiment	Results	Document the results (i.e., measures and metrics) and the analysis.
	Analysis	Explicitly indicate whether the analysis supports the hypotheses.
	Justification	Validate that the chosen datasets, empirical design, and metrics are appropriate for assessing the results.
	Workflow	Workflow representation that summarizes how the experiment is executed and configured.
	Workfl. exec	Workflow execution traces providing settings and initial, intermediate, and final data.
	Hardware	Document the hardware used for running the experiments.
	Software	Document the software dependencies.
	Exp. cite	Automatically generate reference entry for experiment.
	Code repo	Shared code through community repository.
	Code metadata	Include basic metadata for describing the code (language etc.).
	Code license	Include a license.
	Code citeable	Generate a digital object identifier (DOI) or persistent URL (PURL) for the version used.
Method	Hypothesis	Document the hypotheses to be assessed.
	Prediction	Document the predicted outcome of the experiment.
	Setup	Parameters and the conditions to be tested and desired statistical significance of results.
	Prob. desc.	Support description of the problem that is intended to solve.
	Outline	Support for outlining the method conceptually.
	Pseudo code	Support for describing the AI method as pseudo code.
Data	Data repo	Share data in a community repository.
	Data metadata	Include basic metadata that describes the data.
	Data license	Give the data a license.
	Data citeable	Generate a digital object identifier (DOI) or persistent URL (PURL) for the version used.

$$R3F(p) = Method(p), \tag{3}$$

where $Method(p)$, $Data(p)$ and $Exp(p)$ are weighted means of the variables describing the three factors Method, Data and Experiment for a platform p. Hence, a platform p can be scored on every variable based on whether it has features that covers the functionality of each variable. In this way, the metrics can be computed for each platform and the platforms can be compared.

The idea behind the different levels is described by Gundersen and Kjensmo [11]. In short, the more detail that is provided by the original researchers, the better chance for independent researchers to get the exact same results independently. The higher the $R1F$ score, the more variables are covered. However, if only the scientific report is released, independent researchers could still reproduce the results, but not exactly. The higher the $R3F$ score is the easier it is to reproduce results without any code and data. For example, if independent researchers implement an algorithm described in a scientific report and run it on a different set of data, the claims of the original researchers can still be supported although the exact performance metrics will not produce the exact same values. An example could be independent researchers implementing an artificial neural network as described by the original researchers and testing it on a different data set than what the original researchers used. This new implementation could still perform significantly better than some reference method, and hence the result would be reproduced, although a performance measure, such as accuracy or F1 score, would not get the exact same score as the original experiment.

The weights of the factors are δ_1, δ_2 and δ_3 respectively. It is of course possible to give different weights to each variable and factor, but we use uniform weights, $\delta_i = 1$, in our study. One could easily argue that uniform weights do not make sense, as some variables clearly are more important than others for enabling reproducibility. The proposed method illustrates how platforms can be scored using the features suggested in related work without trying to give an answer to which factors and variables are most important. Choosing weights without a very structured or well-argued method for doing this could be disputed. A good set of weights could be a question of policy or based on empirical evidence for which features actually are most important for reproducing results. Our position is that finding the right set of weights could be a research project on its own, and this is not the project we report here.

IV. Research Method

We have assessed 13 machine learning platforms based on the quantitative method proposed above[3]. The platforms have been chosen based on reviewing literature on reproducibility. In addition, we have included the most commonly used machine learning platforms provided by Amazon, Microsoft and Google. The reason we added these machine learning services is that it allows us to analyse whether the more reproducibility

[3]See here for code and data: https://github.com/kireddo/escience2019

oriented platforms provide value in this regard compared to the platforms used daily by the industry.

We use machine learning platform in a broad manner. We do not restrict it to be a cloud solution where machine learning experiments can be executed, solutions that could be installed and run on a local machine are included. The idea is that the platform supports and simplifies developing machine learning programs and provides a rich set of functionality. Platforms include more functionality than a library, such as SciKit-learn[4]. Some of these solutions, such as Polyaxon and Azure ML identify as platforms. StudioML identifies as a framework while OpenML identifies as an environment.

Each platform is scored based on whether a feature is *Supported*, *Partially supported* or *Not supported*. A supported feature gets a score of 1, while a partially supported feature has been scored as 0.5 and a not supported feature is scored 0. We could have used the whole range between 0 and 1 to describe whether a feature is supported, but this would result in a very subjective score, which we wanted to avoid. Partially supported features could either be partially supported as part of the machine learning platform or it could be supported through integration with third party software. In order to get a score of partially supported for integration with third party software the software platform must support such integrations actively.

The primary target for our data collection has been the documentation that is available for each platform. Further investigations have proven necessary in some cases where documentation has been unclear. This has not consisted in properly conducting complete experiments, but rather isolating features where the documentation did not seem to provide sufficient information. Some of the issues with this kind of data collection is discussed further in section VII. The following subsections describe the three factors in more detail.

A. Experiment

The factor Experiment covers the parts of the research which are implemented in software. This includes any novel machine learning methods, the workflow of the experiment, as well as the environment the experiments are executed in. The results can be presented in different ways, such performance metric scores in tables or visualized as graphs. The setup of the experiment should specify and store hyperparameters and environment variables in a understandable representation that can be reviewed later. Workflows are typically represented as graphs where sub-processes of the machine learning experiment are specified as well as the flow of inputs and outputs.

In cloud based computing systems, hardware specifications are typically given as part of the cluster configuration. However, this does not necessarily make it easier for the user to specify and document hardware, as the exact hardware is chosen upon run-time by the cloud platform. None of the assessed cloud computing platforms have ways in place to automatically track the actual hardware (the exact physical machine) used to run experiments. This is contrasted by the careful documentation of equipment taught in undergraduate physics classes. Here, students must document serial numbers of the tools used in order to be able to distinguish between sloppiness and bad tools. In computer science, running on GPUs from different vendors and even different production batches of the same GPU can yield different results [9], so being able to track the exact hardware is clearly valuable. Software dependencies are usually available, often through the use of containers and systems like Docker.

[4]http://scikit-learn.org

B. Method

The factor Method specifies variables that are part of the textual documentation, the scientific report that is written for independent researchers, so that they are enabled to conduct the experiments themselves. It is written by researchers to convey the ideas and concepts behind the research to other researchers. The documentation describes the machine learning method and the experiment setup with hyperparameters and the environment, so that independent researchers understand the reasoning behind performing an experiment in a given way. For example in the context of a scientific paper, it makes sense to present pseudo code as part of the textual description of the method, rather than with the code, as the pseudo code is there to help other researchers understand the algorithm that is presented. As mentioned, notebooks are judged by many as a solution for running reproducible experiments and can reasonably be expected to improve communicating experiments to other researchers to some extent. However, notebook provide free form text, and therefore they do not provide structure for what exactly to document. Because of this, notebooks can only partially satisfy the the factor method at best.

C. Data

The factor Data specifies variables related to whether data is shared and whether it is specified as which samples are used for training, validation and testing. For experiments to be $R1F$ and $R2F$ reproducible, data has to be shared. Hence, in order for a platform to score well on this factor it must offer a possibility to openly host data and tracking which samples were used for what. The most common practice is hosting data on network storage such as S3, Google storage or Azure Blob storage, or on local servers. However, these do not provide versioning, structured meta data, possibility for citing the data or provide licenses. An alternative is to rely on an external data repository. These typically provide more features such as metadata and licenses, as well as Persistent Uniform Resource Locators (PURLs) or Digital Object Identifier (DOIs).

V. Surveyed Platforms

This section provides an overview of the software platforms that have been surveyed.

OpenML[5]: Open source experiment database for machine learning. The platform hosts open data, and defines algorithms in a representation called flows. Datasets and tasks hold rich

[5]https://docs.openml.org/

metadata, and results from tasks are aggregated and compared over different flows. There is no option to private data or experiments, and code has to be run locally and uploaded through one of their APIs. At the time of the survey, the study feature of OpenML was still not fully implemented. This feature is meant to handle the most of the scientific method tied to the experiments. Because this is not fully implemented, the platform has insufficient support for most of the features relying on this.

MLflow[6]: Machine learning framework that is developed by Databricks, currently in beta. The MLflow project is open source and is made to easily integrate with other systems. It is naturally compatible with other systems developed at Databricks. This allows us to access features such as databricks notebooks. The main features of MLflow itself are its experiment tracking, packaging and deployment support.

Polyaxon[7]: Platform made for building, training and monitoring large scale deep learning applications, currently in beta. It is made to support most popular deep learning frameworks and machine learning libraries. Polyaxon requires a Kubernetes cluster to be run. It offers its own tracking UI for experiments.

StudioML[8]: Framework for managing sharing and reproducing Python experiments. It is an attempt to simplify and speed up the machine learning pipeline. The system attempts to avoid being invasive, and should run with little to no alterations to any working python machine learning code. Artifacts, data and logs are stored and organized in predefined data storages.

Kubeflow[9]: Kubernetes native open source machine learning platform, developed at Google. One of the aims is to have a low bar for entry, but a high ceiling for advanced users. Extensive knowledge about Kubernetes should not be necessary for most users. The platform is still in development, and new features are expected to be added in the future. At the moment, the system deployment is built around Ksonnet and TF-serving. Several other projects are also supported.

CometML[10]: Python based machine learning platform for tracking and sharing experiments. One of the interesting features offered by CometML is the ability to compare experiments side by side. This allows easy comparisons for differences in code, convergence and hyperparameters among other things. Documentation can be attached to experiments in form of notes, graphs and charts, making them easier to understand and reproduce.

Amazon Sagemaker[11]: Machine learning platform developed by Amazon, made to run on the Amazon Web Services (AWS). It is built from a few separate parts that can be used independently from each other. The system is built on docker containers, which are used to define the setup of the experiments. There are many available containers supporting different machine learning libraries, and one can also write containers that support custom code.

Google Cloud ML[12]: Google cloud ML engine is a machine learning service built on the Google Cloud Platform (GCP). It supports multiple machine learning frameworks and is integrated with Google storage and Google cloud. It offers a series of custom APIs which are specialized at anything from speech to image recognition. The APIs are packaged separately, so users can pick and choose the features that are desired for their specific systems.

Azure ML[13]: Machine learning platform, developed by Microsoft. It has two different services: service and studio. Azure ML service is a more typical platform for development and deployment that requires users to be able to program, while Studio is a simplified drag and drop tool that builds on the same system. Studio is a good option for scientists who are not machine learning experts.

Floydhub[14]: Commercial machine learning platform for Python experiments. It offers a web dashboard with a number of popular features such as Jupyter and Tensorboard integrated. Floydhub is integrated with Github and offers version control and sharing for both code and data. The platform is built on offering cloud services.

BEAT[15]: Open source machine learning platform, developed at Idiap Research Institute in Switzerland. BEAT hosts both data and source code openly on the platform, but there are also features for hiding experiments, data and code. The platform is built on a component called toolchains, which describes the workflow of the experiments in block diagrams [27].

Codalab[16]: Open source machine learning platform for researchers, built by Microsoft. The platform is split into two parts, competitions and worksheets. The worksheets is the part that primarily looks to support reproducible experiments. Codalab hosts data and source code and offers an interface for easily accessible executable papers.

Kaggle[17]: Data science platform built around sharing of data and machine learning competitions. Kaggle hosts a large data repository, as well as code in the form of notebooks and scripts. The platform offers a free cloud computing service with options to run on both CPU and GPU. The APIs also allow users to easily download content to work on it locally. There is an established community of researchers who frequent Kaggle competitions.

VI. Results

We found that scoring was difficult at times. One challenge in particular is when a platform integrates with some external system like a source control management system or notebooks. One question is whether the integration with the external system fully extends the functionality of the

[6]https://www.mlflow.org/docs/latest/index.html

[7]https://docs.polyaxon.com/, Version 0.2.9

[8]http://docs.studio.ml/en/latest/index.html

[9]https://www.kubeflow.org/docs/about/kubeflow/, Version 0.3

[10]https://comet-ml.com/docs/, Version 1.0.31

[11]https://sagemaker.readthedocs.io/en/latest/, Version 1.11.2

[12]https://cloud.google.com/ml-engine/docs/

[13]https://docs.microsoft.com/en-us/azure/machine-learning/service/

[14]https://docs.floydhub.com/, Version 0.11.14

[15]https://www.beat-eu.org/platform/static/guide/

[16]https://github.com/codalab/codalab-worksheets/wiki

[17]https://www.kaggle.com/docs/kernels

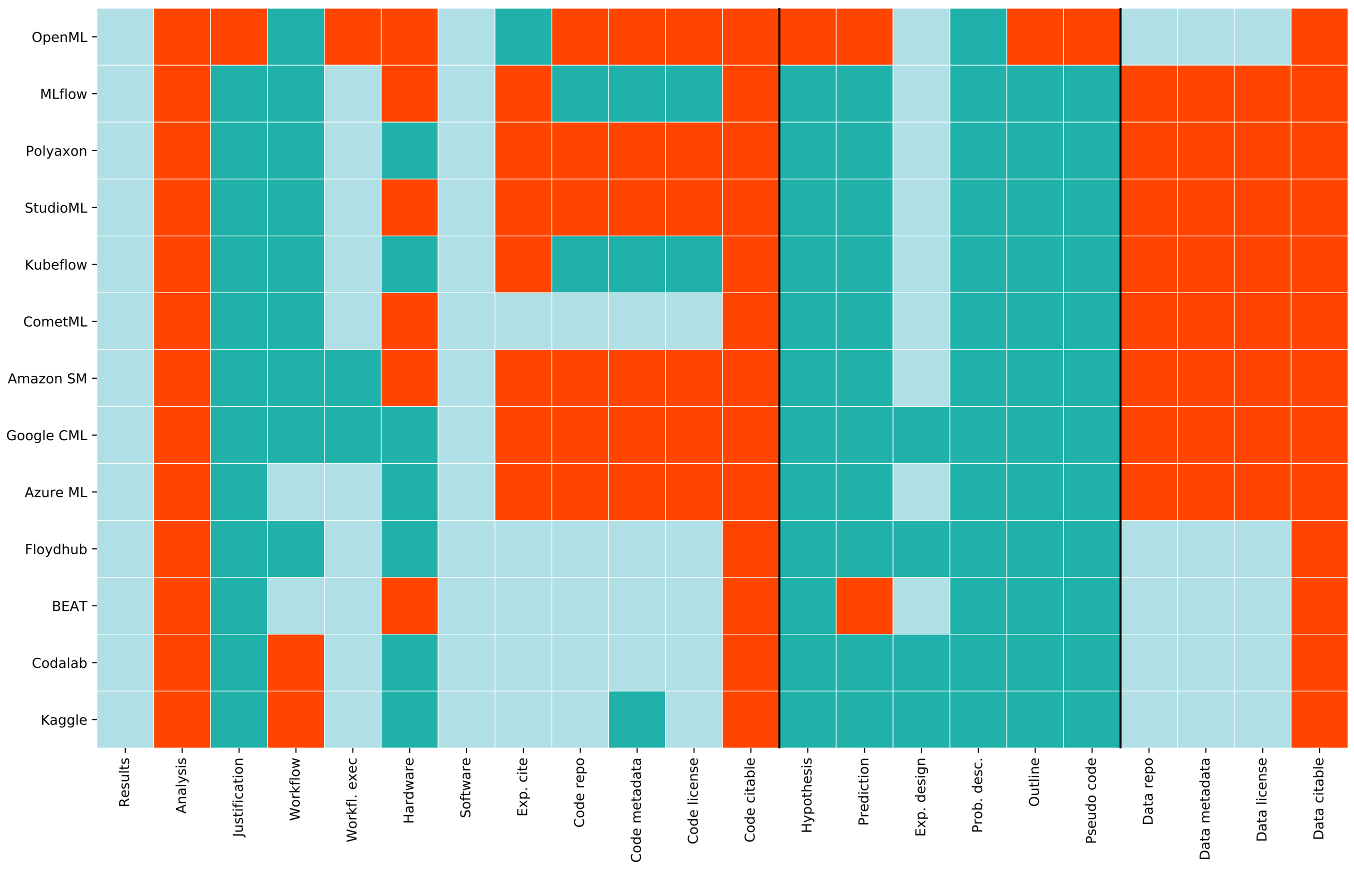

Fig. 1. Heat map showing which software platforms (rows) have the specified features (columns). Light blue indicates that the feature is supported, sea green indicates partially supported and orange that it is not supported.

machine learning platform. Another is whether the integration is actively supported or whether the support is more done as an afterthought, so that the usage feels unnatural. Notable external systems we found platforms using are Notebooks, Git, Tensorboard and docker. The solutions developed by the big tech companies Amazon, Google and Microsoft, integrate with already established infrastructure part of their cloud services, such as file storage and databases.

The results of the survey are illustrated in the heat map in figure 1. The vertical black lines in the figure divide the variables into the three factors, which shows that different platforms typically support use cases that align with the factors.

The heat map shows that all systems lack functionality for the variables data and code citation as well as analysis. Most of the systems also lack functionality for sharing code and data. Sharing code and data are features that typically are related to publishing research, and hence they are not necessarily important features for the commercial machine learning platforms developed for commercial businesses who typically do not want to share code and data. Floydhub, BEAT and Codalab are developed for researchers in order to support reproducibility, and they support data and code sharing natively. Kaggle is mainly a platform for conducting and participating in machine learning competitions, and is therefore build for sharing code and data. Generating permanent URLs and making data sets and code citeable can be done using external services like Zenodo[18], Figshare[19], W3ID[20] and Datacite[21].

Table II displays the external systems that are supported by the the surveyed platforms. These external systems include notebooks (ex. Jupyter), source code management (SCM) systems (ex. git and Github), Docker and Tensorboard. The support of external systems extend the desired functionality of the machine learning platforms not only by providing new functionality, but also supporting the same functionality in a new way. An example is code sharing, which can be provided as part of the the platform, but also through integrating with external repositories. Table III shows which variables are covered by the external systems we identified.

Notebooks: The integration with notebooks turned out to be particularly noticeable. As discussed earlier, a lot of

[18]https://zenodo.org
[19]https://figshare.com
[20]https://w3id.org
[21]https://datacite.org
[22]Source control management system such as Github.

TABLE II
CHECK MARKS INDICATE WHETHER A PLATFORM SUPPORT ONE OF THE EXTERNAL SYSTEMS AND A DASH INDICATES THAT IT DOES NOT. N/A INDICATES THAT WE DID NOT FIND ANHY INFORMATION ABOUT THIS.

	Notebooks	SCM[22]	Docker	Tensorboard
OpenML	-	-	✓	-
MLflow	✓	✓	✓	-
Polyaxon	✓	-	✓	✓
StudioML	✓	-	✓	✓
Kubeflow	✓	✓	✓	✓
CometML	✓	✓	N/A	✓
ASM	✓	-	✓	✓
GCML	✓	-	✓	✓
Azure ML	✓	-	✓	-
Floydhub	✓	✓	✓	✓
BEAT	-	-	✓	-
Codalab	-	-	✓	-
Kaggle	✓	-	✓	-

TABLE III
SHOWS THE VARIABLES THAT ARE COVERED THROUGH INTEGRATIONS WITH EXTERNAL SYSTEMS

System	**Variables supported**
Notebooks	Justification, Hypothesis, Prediction, Problem Description, Outline and Pseudo code
SCM	Code repository, Code metadata and Code license
Docker	Software dependencies
Tensorboard	Workflow, Results

the variables tied to the experiment and method are most easily satisfied through textual descriptions. This means that interfaces that allows the user to attach additional notes to the experiments will contribute a lot to the overall score of a platform. A number of platforms have notebooks as their only option for supporting this kind of documentation.

Among the platforms that integrate with notebooks (see table III), Floydhub is an exception. It allows for documentation be added in the experiment notes, but also offers the use of Jupyter notebooks, which can serve the same purpose. The ability of attaching notes to the experiments is also present in BEAT, Codalab and CometML. Polyaxon, StudioML, Kubeflow, Amazon SM, Google CML, Azure and Kaggle depend on notebook integrations to support the Method variables. The specific variables that are affected by these integrations are justification, hypothesis, prediction, experiment design, problem description, outline and pseudo code. To which degree these variables are supported could be debated, as the integration makes it possible to document the Method variables, but nothing more. As mentioned above, in general, we chose to assign partially supported when this was the case.

Source code management: Only MLflow, Kubeflow, CometML and Floydhub rely on integration with external systems for sharing code by integrating with Github. MLflow and Kubeflow (Argo CD) do not support open sharing of code. The support for license and code metadata are outsourced to Github, which only support this through allowing users to add files that contain this information.

Sharing of code and data brings a set of challenges with it. It is fair to assume that most developers will be versioning their code through some repository already. This means that any feature covering the same functionality will need to meet at least the same standards as what is already being used. Hence, integrating with common source control management systems is an advantage as users know how these work.

Software dependencies: Docker documents software dependencies and is implemented to some degree by most of the surveyed platforms, as a way of dealing with software dependencies. There is however a difference between how much direct interaction the user has with docker. Some of the platforms allow more freedom in using custom docker images, while others offer a selection of already installed images. We could not find information in the documentation for CometML about whether they use Docker for software dependencies, but tracking of software dependencies is supported.

Workflows: Workflows are most easily represented as graphs, and this is how Tensorboard support workflows. Graphs illustrating the workflows are automatically generated in Tensorboard, and this is a very good solution as it reduces manual work and the possibilities for errors. Tensorboard is integrated with several of the platforms, as illustrated in table II. Workflows are represented in other ways as well, and there is a variety in how this feature is implemented. Toolchains that is a part of the BEAT-platform represent workflows as block diagrams that have to be manually specified by the users. Text is also used for specifying Workflows in OpenML and CometML, and it is generated automatically. However, this requires deeper insight into the how the framework operates, and often the text output was massive and almost impossible to interpret.

Execution traces seem to be preserved by default for most of the systems, or at least be possible to preserve for the majority of the platforms. Amazon Cloudwatch and Google Stackdriver are examples of more advanced implementations of execution traces that allows for monitoring and alerts.

Data repositories: None of the surveyed platforms integrate with dedicated data repositories. Data is typically stored on servers or on cloud storage that are not intended for sharing data. Only Floydhub, BEAT, Codalab and Kaggle implement data sharing features.

Hardware specifications: The support for hardware specification is tied to the features provided by the cloud computing platforms. Kaggle, Floydhub and Codalab offers their own already configured machines. The hardware specification can be found within the appropriate systems documentation. Among these, Floydhub is the closest to a satisfactory solution, with multiple hardware options, and documentation that makes it relatively easy to pin down the specifications. We would still argue that the support is partial, as the information should be more detailed. Polyaxon, Kubeflow, Azure and Google CML all allow the configuration of clusters with Kubernetes. This is information that could and should be added automatically, but in some cases it is not.

Experiment citation: Experiment citation is one of the variables where we see the most variance between different platforms. The degree of support is largely up for interpre-

TABLE IV
MEAN OF THE VARIABLES OVER EVERY CATEGORY FOR EACH SYSTEM

Platform	**Experiment**	**Method**	**Data**
OpenML	0.25	0.17	**0.75**
MLflow	0.42	**0.58**	0.00
Polyaxon	0.38	**0.58**	0.00
StudioML	0.33	**0.58**	0.00
Kubeflow	0.50	**0.58**	0.00
CometML	0.67	**0.58**	0.00
Amazon SM	0.29	**0.58**	0.00
Google CML	0.33	0.50	0.00
Azure ML	0.42	**0.58**	0.00
Floydhub	**0.71**	0.50	**0.75**
BEAT	**0.71**	0.50	**0.75**
Codalab	0.67	0.50	**0.75**
Kaggle	0.63	0.50	**0.75**

TABLE V
REPRODUCIBILITY METRIC SCORES FOR THE 13 PLATFORMS.

Platform	**R1F**	**R2F**	**R3F**
OpenML	0.39	0.46	0.17
MLflow	0.33	0.29	**0.58**
Polyaxon	0.32	0.29	**0.58**
StudioML	0.31	0.29	**0.58**
Kubeflow	0.36	0.29	**0.58**
CometML	0.42	0.29	**0.58**
Amazon SM	0.29	0.29	**0.58**
Google CML	0.28	0.25	0.50
Azure ML	0.33	0.29	**0.58**
Floydhub	**0.65**	**0.63**	0.50
BEAT	**0.65**	**0.63**	0.50
Codalab	0.64	**0.63**	0.50
Kaggle	0.63	**0.63**	0.50

tation, as all the platforms that share the experiments openly support this to some degree. The clearest cut is platforms that has options to openly publish experiments against platforms that do not. Even if the private experiment can be shared with specific users on demand, the option to openly publish is primarily what we have looked for. That being said, the option of being able to share private experiments with only specific individuals do have its benefits, but they are less tied to reproducibility. There are platforms that do offer open sharing, but still have been assigned partially support. For example in the case of OpenML, this stems from citation of the experiment as a whole requires multiple links. This is likely to be resolved in the future with the addition of the studies feature, which did not work properly when we tested it. It is important that the experiment is easy to navigate, and gives a good overview, if it is to be cited.

Table IV shows the mean of the variables defining the factors for each of the systems. We can see that only five of the thirteen surveyed platforms score more than zero for all the factors. Table V lists the scores for each system on the three metrics that were defined above. The support for the data factor has a particularly large impact on the total score of R1 and R2, as so many systems lack this.

Figure 2 shows a scatter plot where each of the systems have been plotted with their R1F score on the x-axis, the R2F score on the y-axis and the size of the dot is scaled based on the R3F score. It shows three clusters, where the three

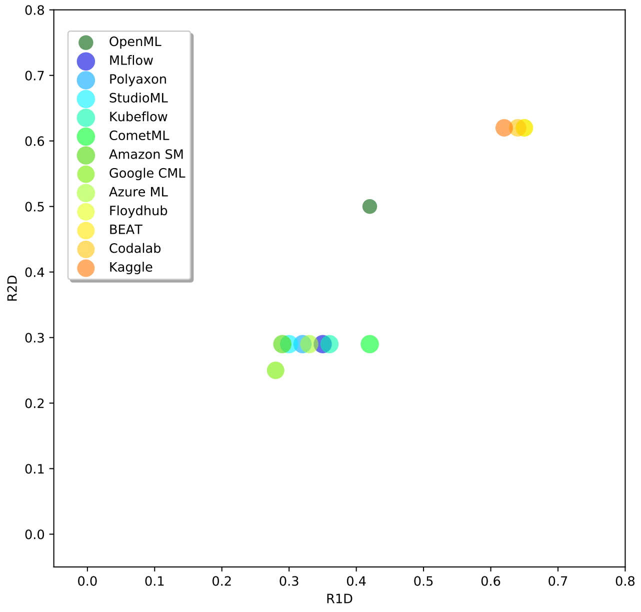

Fig. 2. Plot of platforms where R1 score is on the x-axis, R2 score is on the y-axis and R3 is represented by size.

platforms that have been developed to support reproducibility of machine learning results are in the cluster to the top right together with Kaggle. This cluster represent the leaders while the cluster down and left contains the generalist platforms that do not focus specifically on reproducibility. They still lack lots of functionality to fully support reproducible experiments. Curiously, the central cluster only contains one platform, which is OpenML. OpenML would be among the leading platforms if it had integrated with a source control management system and a notebook. Therefore, with some effort OpenML could be a leading reproducibility platform.

VII. DISCUSSION

The heat map shows that the machine learning platforms clearly support running *experiments*, as most of them support or partially support most variables comprising the factor experiment. This is no surprise as they are developed for this exact use case. Most of these systems also support the factor method to some degree, but this is typically through integration with some sort of notebook, such as Jupyter. Hence, this factor is mostly partially supported. Only OpenML, Floydhub, BEAT, CodaLab and Kaggle of the machine learning systems support the factor Data. This factor is mostly covered by data and/or code repositories only.

What is striking is that none of the platforms that are most commonly used by industry and academia, such as the offerings by Amazon, Google and Microsoft, support the three factors that comprise reproducibility. This means that the results generated by software systems that are being developed for research and commercial products with a high probability are not reproducible.

One aspect that we have not assessed is how practical and user friendly it is to use the different systems. Usability is not easy to quantify and is also subject to subjectivity. What is deemed usable for some are not deemed usable for others. While a platform can technically fully support reproducibility according to this method and analysis, it is possible usability is lost in the way the variables are implemented as features. Some of the systems provide high flexibility in configuration while others do not. Flexibility is also a characteristic of a system that will attract some power users, but for others this only adds complexity and will thus repel them. Both flexibility and usability have not been assessed as these characteristics are hard if not impossible to quantify in an objective manner.

An aspect that is not properly reflected by the factor data is the level of which data can be supported. Limitations tied to the size and type of data that can be stored in a data repository can potentially render the features obsolete if they do not match the required standard of the experiment. If the data is to be updated at any point, data versioning is also important. Data very often change over time. Some data is pre-processed and the pre-processing method might change and with it the data. Also, samples may be added or removed, which is often the case, also in published data sets.

It is important that the platforms do not get in the way of development machine learning systems, as the main reason professionals use them in the first place is to increase the efficiency of the machine learning pipeline. The platform should support an improve the development of machine learning systems. The value of machine learning platforms should not only be to document the experiment for colleagues and independent researchers in order to enable reproducibility. It is important that the value of using the systems outweigh the cost of setting them up for the end user. Keeping track of workflows, which performance metrics are used as well as hyperparameters, which data and code were used are all important parts of the development process, as they support identifying hard-to-find machine learning bugs. According to Irakli Loladze, a matematical biologist, ensuring that the research is reproducible increases time spent on a project by 30% [1]. How these numbers compare with the numbers for computer science is not clear. However, by developing software solutions that support reproducibility, we should be able to reduce the overhead by adding features that takes care of the reproducibility automatically.

Code very often change hands in the industry while this is not necessarily the case for researchers. It is not necessarily the same person who creates the model who will deploy it and later maintain it. Companies do not work on static data sets, but their data sets typically changes all the time. Also, performance of machine learning models often have a direct correlation to the earnings of a company. Hence, making sure the results of experiments are reproducible is something that should be an important aspect of machine learning systems developed for the industry. Surprisingly, the platforms that are developed for and used by the industry do not support reproducibility very well.

Deployment and serving of the created models are also important parts of commercial machine learning projects. Several of the surveyed platforms, particularly the larger commercial ones focus more on this. This feature was not taken into account in the survey, as it is not relevant for reproducing the results. Good software development practices are important for industry, also when it comes to machine learning systems, as these are considered as the high interest credit cards of software development [28].

The majority of the reviewed systems are still in development, some even in alpha. This means that the likelihood of more features being added is rather high. Several of the systems are also open source, allowing for a larger community to improve them. None support persistent URLs, which have been recommended in several papers. The main reason for this is probably that this is a niche feature, as even researchers do not often share data.

There are many challenges tied to implementing integrated systems that support reproducibility. The field of AI utilizes many different software tools, which can be difficult to provide up to date support for. There is a large diversity in programming languages and data sets, and results can vary from the smallest change in experimental setup as demonstrated by Hong et al. [8].

Different platforms end up having different use cases, and this context is important when evaluating them. For example, MLflow can be utilized as a component in a commercial machine learning pipeline, but can also be used on its own. It is also fair to note that there are many other features that are not necessarily tied to reproducibility, which are important for scientists making the choice of the toolsthat suits them best. Some of these include: availability, computational efficiency, security, cost, scale, language and library support, ease of use, learning curve and supported systems. The importance of these factors will depend on the user, and is therefore difficult to objectively analyze.

VIII. Conclusion and Future Work

Based on the results of this assessment, BEAT and Floydhub supports reproducible machine learning experiments the best as they have the highest R1F scores at 0.65. Codalab and Kaggle are close with scores 0.64 and 0.63 respectively.

This is clearly shown in figure 2 where these four systems cluster together in the top right corner. These are all systems that are developed to support reproducible results, so this should be expected. The machine learning systems developed by Amazon, Google and Microsoft that have the most users do not compare well with scores 0.29, 0.28 and 0.33 respectively. Future work includes implementing experiments and run them on the top scoring platforms and in this way evaluate usability and flexibility. We would also like to investigate proper weights of variables in the reproducibility metrics.

Acknowledgments

This work has been carried out at the Norwegian Open AI Lab at the Norwegian University of Science and Technology, Trondheim, Norway.

References

[1] Monya Baker. 1,500 scientists lift the lid on reproducibility. *Nature*, 533(7604):452–454, may 2016.

[2] Joelle Pineau. Reproducibility, reusability, and robustness in deep reinforcement learning, 2018. Keynote at ICLR 2018.

[3] Brian A Nosek, George Alter, George C Banks, Denny Borsboom, Sara D Bowman, Steven J Breckler, Stuart Buck, Christopher D Chambers, Gilbert Chin, Garret Christensen, et al. Promoting an open research culture. *Science*, 348(6242):1422–1425, 2015.

[4] Berna Devezer, Luis G. Nardin, Bert Baumgaertner, and Erkan Ozge Buzbas. Scientific discovery in a model-centric framework: Reproducibility, innovation, and epistemic diversity. *PLOS ONE*, 14(5):1–23, 05 2019.

[5] Mikio L Braun and Cheng Soon Ong. Open science in machine learning. *Implementing Reproducible Research*, pages 343–345, 2014.

[6] Christian Collberg and Todd A. Proebsting. Repeatability in computer systems research. *Communications of the ACM*, 59(3):62–69, February 2016.

[7] Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. Deep reinforcement learning that matters. 2018.

[8] Song-You Hong, Myung-Seo Koo, Jihyeon Jang, Jung-Eun Esther Kim, Hoon Park, Min-Su Joh, Ji-Hoon Kang, and Tae-Jin Oh. An evaluation of the software system dependency of a global atmospheric model. *Monthly Weather Review*, 141(11):4165–4172, 2013.

[9] Prabhat Nagarajan, Garrett Warnell, and Peter Stone. Deterministic implementations for reproducibility in deep reinforcement learning. January 2019.

[10] Geir Kjetil Sandve, Anton Nekrutenko, James Taylor, and Eivind Hovig. Ten simple rules for reproducible computational research, 2013.

[11] Odd Erik Gundersen and Sigbjørn Kjensmo. State of the art: Reproducibility in artificial intelligence. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[12] Jonathan B. Buckheit and David L. Donoho. Wavelab and reproducible research. Technical report, Standford, CA, 1995.

[13] Jon F. Claerbout and Martin Karrenbach. Electronic documents give reproducible research a new meaning. In *Proceedings of the 62nd Annual International Meeting of the Society of Exploration Geophysics*, New Orleans, USA, 1992. 25 to 29 October 1992.

[14] Yolanda Gil, Cédric H David, Ibrahim Demir, Bakinam T Essawy, Robinson W Fulweiler, Jonathan L Goodall, Leif Karlstrom, Huikyo Lee, Heath J Mills, Ji-Hyun Oh, et al. Toward the geoscience paper of the future: Best practices for documenting and sharing research from data to software to provenance. *Earth and Space Science*, 3(10):388–415, 2016.

[15] Akshay Sethi, Anush Sankaran, Naveen Panwar, Shreya Khare, and Senthil Mani. Dlpaper2code: Auto-generation of code from deep learning research papers. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

[16] Joel Grus. I do not like notebooks, 2018. Talk at JupyterCon 2018.

[17] Mark D Wilkinson, Michel Dumontier, IJsbrand Jan Aalbersberg, Gabrielle Appleton, Myles Axton, Arie Baak, Niklas Blomberg, Jan-Willem Boiten, Luiz Bonino da Silva Santos, Philip E Bourne, et al. The fair guiding principles for scientific data management and stewardship. *Scientific data*, 3, 2016.

[18] Victoria Stodden, Marcia McNutt, David H Bailey, Ewa Deelman, Yolanda Gil, Brooks Hanson, Michael A Heroux, John PA Ioannidis, and Michela Taufer. Enhancing reproducibility for computational methods. *Science*, 354(6317):1240–1241, 2016.

[19] Joan Starr, Eleni Castro, Mercè Crosas, Michel Dumontier, Robert R Downs, Ruth Duerr, Laurel L Haak, Melissa Haendel, Ivan Herman, Simon Hodson, et al. Achieving human and machine accessibility of cited data in scholarly publications. *PeerJ Computer Science*, 1:e1, 2015.

[20] Chris Drummond. Replicability is not reproducibility: nor is it good science. *ICML workshop*, 2009.

[21] Victoria C. Stodden. Trust your science? Open your data and code. *Amstat News*, pages 21–22, 2011.

[22] Roger D. Peng. Reproducible research in computational science. *Science*, 334(6060):1226–1227, 2011.

[23] Steven N. Goodman, Daniele Fanelli, and John P. A. Ioannidis. What does research reproducibility mean? *Science Translational Medicine*, 8(341):341ps12–341ps12, jun 2016.

[24] Rachael Tatman, Jake VanderPlas, and Sohier Dane. A practical taxonomy of reproducibility for machine learning research. 2018.

[25] Hans E. Plesser. Reproducibility vs. replicability: A brief history of a confused terminology. *Frontiers in Neuroinformatics*, 11:76, 2018.

[26] Odd Erik Gundersen, Yolanda Gil, and David Aha. Towards reproducible research, open science, and digital scholarship in ai publications. *AI magazine*, 39(3):56–68, 2018.

[27] André Anjos, Laurent El Shafey, and Sébastien Marcel. Beat: An open-science web platform. In *Thirty-fourth International Conference on Machine Learning*, August 2017. https://openreview.net/group?id=ICML.cc/2017/RML.

[28] D Sculley, Gary Holt, Daniel Golovin, Eugene Davydov, Todd Phillips, Dietmar Ebner, Vinay Chaudhary, and Michael Young. Machine learning: The high interest credit card of technical debt. 2014.