Future Generation Computer Systems 126 (2022) 34–47

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

Do machine learning platforms provide out-of-the-box reproducibility?

Odd Erik Gundersen *, Saeid Shamsaliei, Richard Juul Isdahl

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

ARTICLE INFO

Article history:
Received 15 March 2020
Received in revised form 7 April 2021
Accepted 8 June 2021
Available online 17 July 2021

Keywords:
Reproducibility
Reproducible AI
Machine learning
Survey
Reproducibility experiment

ABSTRACT

Science is experiencing an ongoing reproducibility crisis. In light of this crisis, our objective is to investigate whether machine learning platforms provide out-of-the-box reproducibility. Our method is twofold: First, we survey machine learning platforms for whether they provide features that simplify making experiments reproducible out-of-the-box. Second, we conduct the exact same experiment on four different machine learning platforms, and by this varying the processing unit and ancillary software only. The survey shows that no machine learning platform supports the feature set described by the proposed framework while the experiment reveals statstically significant difference in results when the exact same experiment is conducted on different machine learning platforms. The surveyed machine learning platforms do not on their own enable users to achieve the full reproducibility potential of their research. Also, the machine learning platforms with most users provide less functionality for achieving it. Furthermore, results differ when executing the same experiment on the different platforms. Wrong conclusions can be inferred at the at 95% confidence level. Hence, we conclude that machine learning platforms do not provide reproducibility out-of-the-box and that results generated from one machine learning platform alone cannot be fully trusted.

1. Introduction

A concern has grown in the scientific community related to the reproducibility of scientific results. The concern is not unjustified. According to a Nature survey, the scientific community is in agreement that there is an on-going reproducibility crisis [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]. 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 [4]. However, even though this is true, computer science and machine learning research is not necessarily reproducible. Collberg and Proebsting report an experiment in which they tried to execute the 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 [5]. In their experiment, they only tried to run the code; they did not evaluate whether the results were reproducible.

Machine learning is still and to a very large degree an empirical science, so the issues with reproducibility is a concern. For example, to establish which algorithm is better for a task, an experiment is designed where the algorithms are trained and tested on the same datasets that represent the task. The one that compares best according to one or more performance metrics is deemed to be the best for a given task. Now, imagine that we have two algorithms that we want to compare. Algorithm A is our own and algorithm B is developed by third party. The results depend on how much documentation that is made available to us by the third party whom authored algorithm B. For example, if we only have access to written material, we have to implement the algorithm ourselves and test it on data that we collect ourselves. There is practically no way we can verify that we have implemented and configured the algorithm in the exact same way as the original authors. So, the more documentation (textual description, code and data) that is released by the original investigators, the easier for independent investigators to reproduce the reported results.

While making more documentation available to independent researchers increases the *reproducibility potential*, reproducibility is not guaranteed. By reproducibility potential we mean the probability of being able to reproduce the results, given that the effect documented by the original experiment is true. True effects can be obfuscated in many ways. For example, the data does not reflect the actual world and hence using a different dataset will

* Corresponding author.
E-mail address: odderik@ntnu.no (O.E. Gundersen).

https://doi.org/10.1016/j.future.2021.06.014

produce different results [6]. Furthermore, baseline implementations produce different results, hyperparameter settings affect the outcomes, and stochasticity in algorithms and environments affect results [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]. Finally, 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]. One of these effects alone can obfuscate the results so much that the conclusions that can be inferred from the experiments are false.

Now, if the two algorithms we want to compare have the exact same performance, will we be able to establish this when the experiments involving each of these algorithms are conducted in different laboratories? This is a relevant question as algorithms often are compared without redoing the experiment with the algorithm developed by a third-party. Instead, results that are reported in a scientific article is used for comparison. Also, independent investigators find that it is hard to get the same results when redoing complete experiments reported in scientific literature. This includes using the exact same code to conduct the exact same experiment using the exact same data. Computer science is in the fortunate situation that the exact same experimental procedures can be followed using the exact same data. The only difference is the laboratory where the experiment is conducted, where laboratory means different hardware and different ancillary software. To evaluate whether it is possible to establish that two algorithms have the exact same performance, the exact same experiment (exact same code and data) can be conducted in different laboratory configurations to produce the same results. In this paper, we follow the definition of reproducibility and of different laboratories (different ancillary software and hardware) that is provided by Gundersen [10].

Goal: Our goal is to find out whether current machine learning platforms support reproducibility out-of-the-box. We do this by investigating (1) how well machine learning platforms enable users to achieve the full reproducibility potential and (2) whether conducting a machine learning experiment in a different laboratory can obfuscate the results to such a degree that the wrong conclusion will be inferred.

Contributions: Our main contributions are fourfold: (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, (iii) we analyze which features that should be developed for the different platforms in order to improve reproducibility support and increase the reproducibility potential, (iv) and, finally, we conduct the exact same experiment a total of 160 times only changing the laboratory configuration to analyze how results differ with the different laboratory configurations.

Results: Machine learning platforms do not provide out-of-the-box reproducibility. The results of the survey show that no machine learning platform supports the feature set described by the proposed framework. Hence, the machine learning platforms do not enable users to achieve the full reproducibility potential without using additional third party software. Our experiment shows that there can be significant differences in results depending on which machine learning platform is used when conducting the experiment. As these results are significant at 95% confidence level, the wrong conclusions can be drawn when effect size is small and the experiment is conducted on one machine learning platform only.

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 survey and the results are presented in Section 4. Section 5 contains a reproducibility experiment based on a simple classification task, its result and a discussion. Finally in Section 6, we conclude and provide some thoughts on future work.

2. 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 [11]. 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 [12]. 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 [13]. 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 [14]. 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 [15]. 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 [16]. In the report by the National Academies of Sciences, Engineering and Medicine [17], reproducibility is defined to mean computational reproducibility and replicability to mean obtaining consistent results across studies answering the same scientific questions. While reproducibility does not necessarily mean discovery of truth, as Devezer et al. [18] suggest, enabling reproducibility makes the analyses and evaluations transparent. Plesser provides a historical overview [19] while Gundersen [10] provides a systematic survey of reproducibility definitions.

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 [20,21]. Gil et al. suggested that the experiment procedures should be made explicit [22]. Sethi et al. even proposed autogenerating code for deep neural network architectures by analyzing research papers and in this way reproducing the results [23]. Executable notebooks, such as Jupyter Lab, have been proposed as solutions for reproducibility [24], but everyone does not agree that they are the silver bullet [25]. Alternatives to Jupyter notebooks exist [26].

In addition to the suggested solutions, several recommendations for combating the reproducibility crisis have been made by Wilkinson et al. [27], Stodden et al. [28], Nosek et al. [3], Gil et al. [22], Starr et al. [29] and several others. Cockburn et al. [30] provide an overview of the threats and potential solutions to these. Remedies for combating the reproducibility crisis (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.

3. Quantifying support for reproducibility

Our work build on the method for quantifying reproducibility that are suggested by Gundersen and Kjensmo [15]. 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. [31] 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 1 for a description of the variables and the factors they belong to.

The three reproducibility metrics are defined as follows:

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

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

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

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 [15]. 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.

4. A survey of the reproducibility support of machine learning platforms

Given that reproducibility is quantifiable according to the described framework, we could investigate how well different machine learning platforms support reproducibility by analyzing whether the platforms support the factors. This can be done by analyzing whether the platforms have functionality that implements each of the variables. The survey is an observational study where we go through the documentation of a set of machine learning platform. The framework could also be used as a requirement specification when developing a machine learning platform that supports reproducibility.

4.1. Research method: Survey

We have assessed 13 machine learning platforms based on the quantitative method proposed above.[1] 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 analyze whether the more reproducibility oriented platforms provide value in this regard compared to the platforms used daily by the industry.

We use the term 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.[2] 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.

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

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

Table 1
Contains the definitions of what the different variables mean, and which factors they belong to.

Factor	Variable	Description
Experiment	Results	Document the results and the analysis.
	Analysis	State how the analysis supports the claims.
	Justification	Justify datasets, method, and metrics.
	Workflow	Workflow representation summarizing experiment execution and configurations.
	Workflow execution	Workflow execution traces with settings raw, processed, and final data.
	Hardware	Hardware used to conduct the experiment.
	Software	Document the software dependencies.
	Citation Export	Automatically generate reference.
	Code repository	Shared code in repository.
	Code metadata	Include metadata for describing the code.
	Code license	Include a license.
	Code citeable	Generate a digital object identifier (DOI) or persistent URL (PURL) for experiment.
Method	Hypothesis	Document the hypotheses to be assessed.
	Prediction	Document the predicted outcome.
	Setup	Parameters and conditions to be tested and desired statistical significance of results.
	Problem description	Description of problem to be solved.
	Outline	Describe method conceptually.
	Pseudo code	Support for pseudo code.
Data	Data repository	Share data in a community repository.
	Data metadata	Include basic metadata describing the data.
	Data license	Give the data a license.
	Data citeable	Generate DOI or PURL.

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 4.4. The following subsections describe the three factors in more detail.

4.1.1. Experiment

The factor Experiment covers the parts of the research that 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 subprocesses 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 broken 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.

4.1.2. 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 factor method at best.

4.1.3. Data

The factor Data specifies variables related to whether data is publicly shared and whether the samples used for training, validation and testing are specified. For experiments to be *R*1*F* and *R*2*F* 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).

4.2. Surveyed platforms

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

OpenML[3]: 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 metadata, and results from tasks are aggregated and compared over different flows. There is no alternative to open data and experiments. Code has to be run locally and uploaded through one of their APIs for sharing. 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[4]: Machine learning framework that is developed by Databricks, at the time of writing this 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,[5] Platform made for building, training and monitoring large scale deep learning applications, and at the time of writing this 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[6]: Framework for managing sharing and reproducing Python experiments. It is an attempt to simplify and speed up the development of 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[7] Kubernetes is a native open source machine learning platform, developed at Google. One of its 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[8] 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[9] 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[10]: 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[11]: 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.

Floydhub[12] 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[13]: 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 [32].

Codalab[14]: 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[15]: 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.

4.3. Results of survey

Some of the variables could not be scored unambiguously. One challenge in particular is when a platform integrates with an external system like a source control management system or notebooks. Does the integration with the external system fully extends the functionality of the machine learning platform? Also, should there be a difference in scoring when the integration is actively supported compared with support being implemented in the system but poorly? We ended up not distinguishing to avoid subjective scoring and scored integrations as partially supporting the functionality.

The results of the survey are illustrated in the heat map in Fig. 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.

3 https://docs.openml.org/
4 https://www.mlflow.org/docs/latest/index.html
5 https://docs.polyaxon.com/ Version 0.2.9:
6 http://docs.studio.ml/en/latest/index.html
7 https://www.kubeflow.org/docs/about/kubeflow/, Version 0.3:
8 https://comet-ml.com/docs/, Version 1.0.31:
9 https://sagemaker.readthedocs.io/en/latest/, Version 1.11.2:
10 https://cloud.google.com/ml-engine/docs/
11 https://docs.microsoft.com/en-us/azure/machine-learning/service/
12 https://docs.floydhub.com/, Version 0.11.14:
13 https://www.beat-eu.org/platform/static/guide/
14 https://github.com/codalab/codalab-worksheets/wiki
15 https://www.kaggle.com/docs/kernels

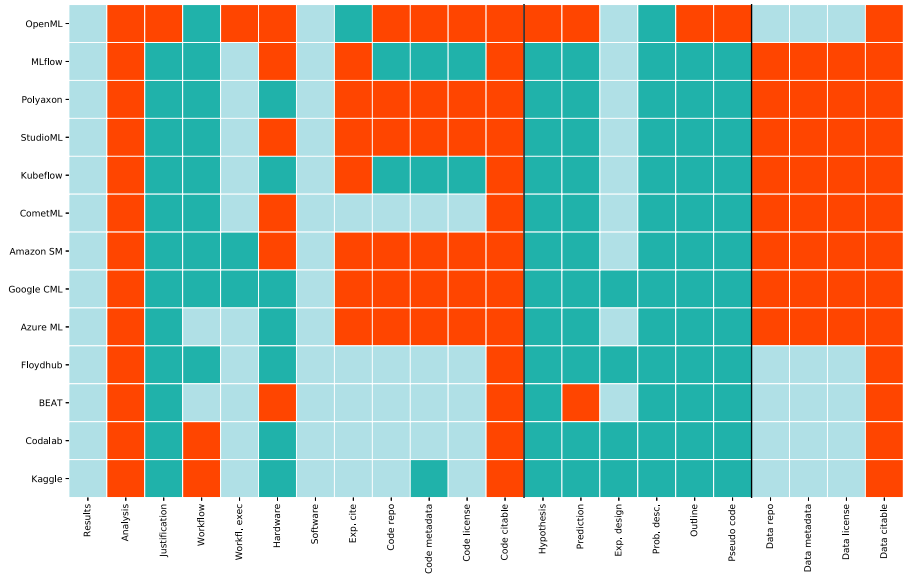


Fig. 1. Heat map showing which software platforms (rows) have the specified features (columns). Light blue indicates that the feature is supported, sea green indicates partially supported and orange that it is not supported.

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 publicly sharing code and data. Publicly 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 to support reproducible research practices, and they support public 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,[16] Figshare,[17] W3ID[18] and Datacite.[19]

Table 2 displays the external systems that are supported by 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 platform, but also through integrating with external repositories. Table 3 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 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 3), 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,

16 https://zenodo.org
17 https://figshare.com
18 https://w3id.org
19 https://datacite.org

Table 2
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 any information about this.

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

Table 3
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

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 2. 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 possible to preserve as default 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 interpretation, as all the platforms that share the experiments openly support this. 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. 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 might be resolved in the future with the addition of the studies feature, which did not work as intended when we tested it.

Table 4 shows the mean of the variables defining the factors for each of the systems. Only five of the thirteen surveyed platforms score more than zero for all the factors. Table 5 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.

Fig. 2 shows a scatter plot where each point refers to a platform with the R1F score plotted on the x-axis and the R2F score on the y-axis. The size of the point is scaled according to the R3F score. There are two main clusters, where the three platforms that have been developed explicitly for enabling reproducibility belong to 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 have not been developed to support reproducible experiments. They still lack lots of functionality to fully support reproducible experiments. OpenML is located alone between the two clusters; it would have been among the leading platforms to the upper right if it had integrated with a source control management system and a notebook.

4.4. Discussion of survey results

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. Most of these systems also support the factor method to some degree, but this is typically through integration with notebooks. Hence, this factor is mostly partially supported. Only OpenML, Floydhub, BEAT, CodaLab and Kaggle of the machine learning systems support the factor Data.

What is striking is that the platforms that have the most users, such as the offerings by Amazon, Google and Microsoft, lack support for reproducibility. However, making sure the results of experiments are reproducible is something that should be an important aspect of machine learning systems developed for the industry and not only for academia. Code very often change

Table 4
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 5
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

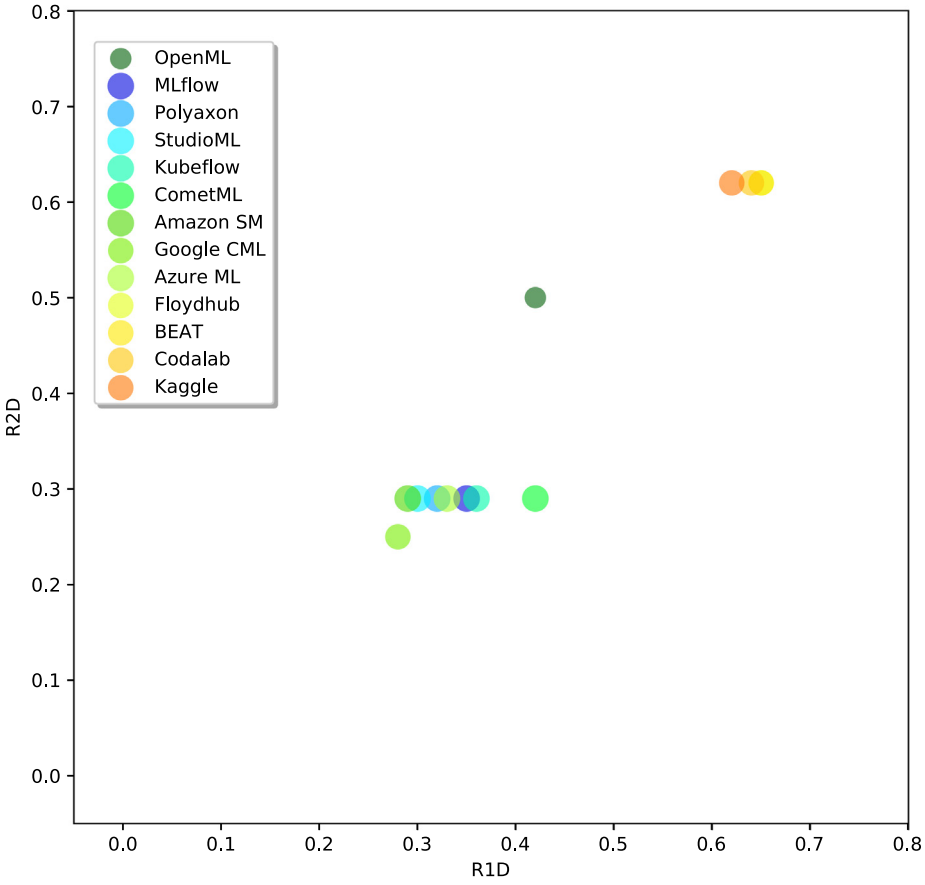


Fig. 2. Plot of platforms where R1 score is on the *x*-axis, R2 score is on the *y*-axis and R3 is represented by size.

hands in the industry while this is not necessarily the case for researchers. It is not necessarily the same person who creates a machine learning 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. 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 [33].

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].

4.5. Conclusion of survey

Based on the results of this survey, BEAT and Floydhub supports reproducible machine learning experiments the best as they have the highest R1F scores at 0.65. Codalab and Kaggle follow closely with scores of 0.64 and 0.63 respectively. The commercial machine learning platforms 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. This means that most experiments are not reproducible unless special care is taken.

5. The reproducibility of digits classification

In this section, we investigate the reproducibility of a machine learning experiment when it is executed on different machine learning platforms. Here, our main goal is to find out whether the results are the same when the exact same experiment is executed on different machine learning platforms. The machine learning platforms run as services in the cloud, so we do not control the hardware, compiler settings or the operating systems of the environment in which the experiments are executed. This is how most machine learning experiments are conducted for most researchers and practitioners. We will analyze the results in four different settings where we vary between fixing and not fixing the seeds for the pseudo-random number generator and when executing the experiment on either the CPU and GPU. Our secondary goal is to find out whether a variation in results will affect any conclusions we can draw from the evaluations of results conducted on different machine learning platforms. The null hypothesis is that we will get the exact same results on all experiments. However, given what we know about how floating point calculations affect weather simulations [8], how running experiments on the GPU affects the determinism of results [9] and how the stochasticity of modern machine learning techniques, such as deep neural networks, affect the results [7], we do not predict the results to be exactly the same. Thus, we predict the null hypothesis to be rejected. This means that we do not expect the results to be outcome reproducible. We do however expect the experiments to be analysis reproducible [10], meaning that although each experiment produce different outcomes, performing the same analysis on the outcomes will still lead to the same conclusions.

5.1. Experiment methodology

The task is to classify images of handwritten digits from the MNIST dataset.[20] The MNIST dataset consists of 60,000 28×28 grayscale images of handwritten digits along with a test set of 10,000 images. It is provided by many classification libraries, such as Keras [34], as one of several standard datasets to evaluate algorithms on. We train a simple and standard convolutional neural network (CNN) as the classifier. It easily reaches a classification error (misclassified instances divided by the total number of instances multiplied by 100) on the test set as low as 1%. Interpreted in this setting, the null hypothesis is that all images are classified as the same class independent of which machine learning platforms the experiment is conducted on.

We chose the machine learning platforms to compare based on their ranking in the survey, and we chose to conduct this experiment on the five highest scoring platforms (BEAT, Floydhub, Kaggle, CodaLab and CometML). However, after quite some trial and error, we decided to drop BEAT from the experiment, as we were not able to run the experiment on the platform. We did not add a new platform for the reproducibility experiment.

As we varied the configurations between fixing the random seed and not fixing it and we conducting them both on the CPU and the GPU, we had 16 different configurations in total (4 platforms with 4 configurations each). For each of the configurations we trained the deep convolutional network with the training set to perform the classification task on the test set five times. In total the classification task was performed 80 times using the same training and test datasets. Every time we performed the classification task, we classified each of the 10,000 images in the test set. This means that we have classified each image in the test set five times for each configuration. For each configuration, the class that was given to an image was recorded every time the experiment was executed.

The experiment was implemented using TensorFlow 1.9, but after having conducted the whole experiment and analyzed the results, we found that there had to be a bug in the software. Therefore, we conducted the experiment on TensorFlow 2.4 as well to see if the bug was fixed and how much it had affected the results. All code implementing the experiments are shared on Zenodo [35].

5.2. Empirical results

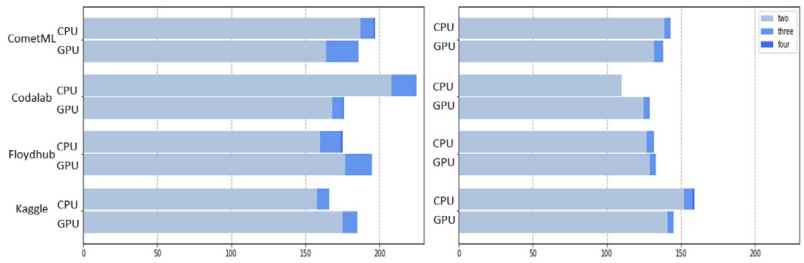
The results are presented in one table and three figures, and the ANOVA is shared with the code in the supplementary materials [35]. Table 6 lists the mean error rate with 95% confidence interval. Fig. 3 and 4 illustrate the number of models that misclassify the images and how many different classes the models misclassify the images as for TensorFlow version 1.9 and 2.4 while Fig. 5 shows how the different platforms rank on the different configurations.

The following observations are made:

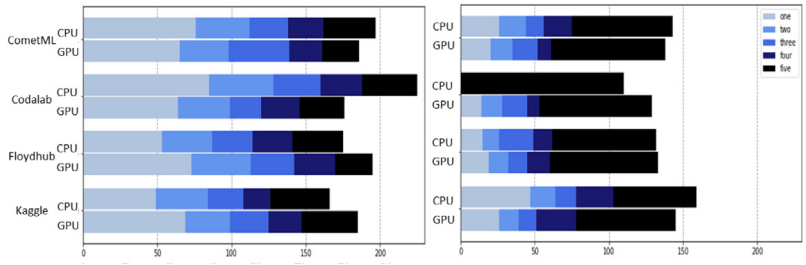
Mean classification error: As can be seen in Table 6, fixing the random seed on TensorFLow 2.4 leads to the expected outcome, that is no variation of the results when conducting the experiment on CPU. The exception is Kaggle where one of the experiments switches more or less half of the time between 86 and 87 misclassifications. We tried several times, but always with the same result. Comparing results between platforms, processing units and TensorFlow versions not taking CPU fixed seed into account, examples can be found where there is significant differences in results between the configurations.

Analysis of variation: Even though the mean classification error and standard deviation vary for all the experiments, they could be drawn from populations with the same mean values. To evaluate this, we conducted both a one-way and a two-way analysis of variance (ANOVA) with platform and

[20] http://yann.lecun.com/exdb/mnist/

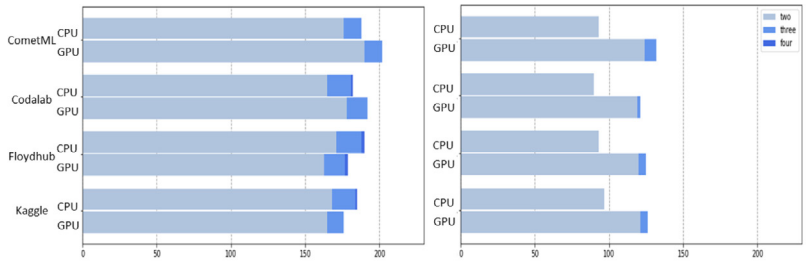


(a) Number of different classes when not fixing the seeds (left) and when fixing the seeds(right).

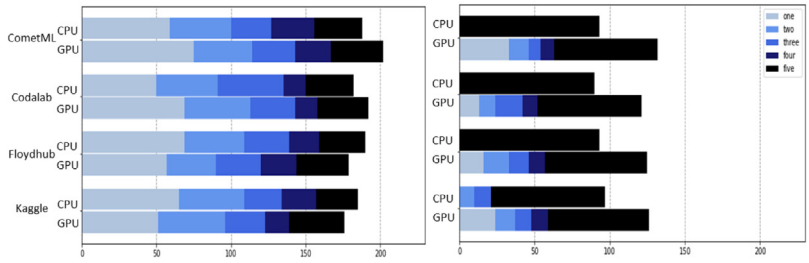


(b) Number of times test images are classified incorrectly in 1, 2, 3, 4 or 5 experiments among 5 experiments, when not fixing then seeds (left) and when fixing the seeds (right).

Fig. 3. illustration of the results of Tensorflow 1.9.



(a) Number of different classes when not fixing the seeds (left) and when fixing the seeds(right).



(b) Number of times test images are classified incorrectly in 1, 2, 3, 4 or 5 experiments among 5 experiments, when not fixing then seeds (left) and when fixing the seeds (right).

Fig. 4. illustration of the results of Tensorflow 2.4.

TensorFlow version as the independent variables (factors) and the number of misclassifications as the dependent variable. We found that the mean classification error is significantly higher for TensorFlow version 1.9 when experiments were conducted on GPU with fixed random seed ($p \ll 0.001$ by two-way ANOVA). Fixing the seed on GPU does not have the same effect on TensorFlow version 2.4 though. Also, the classification error is dependent on which platform the experiment is conducted on when not fixing the seed on the CPU for both versions of TensorFlow ($p < 0.01$ by a two-way ANOVA). A one-way ANOVA on the separate versions of TensorFlow showed that it is the 1.9 version that contributes most to this as the classification error varies most between ($p < 0.01$ by a one-way ANOVA).

Analysis of the misclassification: Fig. 3(a) and 4(a) plot how many different classes an image is classified as on the different platforms for TensorFlow 1.9 and 2.4, respectively.

Table 6
Mean classification error and 95% confidence interval for the experiments conducted on the different machine learning platforms and for the two TensorFlow versions.

TensorFlow version	Platform vendor	CPU		GPU	
		Fixed	Random	Fixed	Random
1.9	CometML	1.02 ± 0.05	0.93 ± 0.03	1.04 ± 0.03	0.99 ± 0.04
	Floydhub	1.02 ± 0.02	0.96 ± 0.06	1.02 ± 0.04	0.95 ± 0.07
	Codalab	1.10 ± 0.00	1.13 ± 0.08	1.01 ± 0.03	0.90 ± 0.04
	Kaggle	1.01 ± 0.07	0.93 ± 0.02	1.06 ± 0.03	0.97 ± 0.04
	ALL	1.04 ± 0.02	1.00 ± 0.05	1.04 ± 0.02	0.95 ± 0.03
2.4	CometML	0.93 ± 0.00	1.00 ± 0.03	0.92 ± 0.02	1.02 ± 0.08
	Floydhub	0.93 ± 0.00	0.94 ± 0.07	0.94 ± 0.01	0.96 ± 0.03
	Codalab	0.90 ± 0.00	0.96 ± 0.08	0.95 ± 0.03	0.95 ± 0.10
	Kaggle	0.86 ± 0.01	0.92 ± 0.05	0.92 ± 0.03	0.94 ± 0.05
	ALL	0.91 ± 0.01	0.95 ± 0.03	0.94 ± 0.01	0.97 ± 0.04

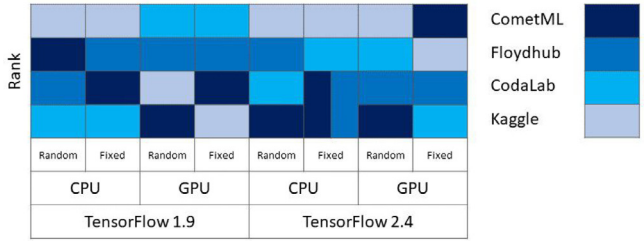


Fig. 5. Ranking of platforms based on the performance on classification task.

As each image is classified five times on each platform with a specific configuration, at most, an image could be classified as five different classes. The first observation is that when models fail to classify correctly, they mostly agree on which wrong class the images belong to. There are only a few examples of four classes and no examples of five. Two different classes is the most dominant outcome when the models disagree. Fig. 3(b) and 4(b) plot how many of the models misclassify a given image when it is misclassified for TensorFlow 1.9 and 2.4, respectively. The figures illustrate that when fixing seeds more models misclassify than when not fixing the seed. Five models being wrong is the most dominant outcome when seeds are fixed. When seeds are fixed on CPU for TensorFlow 2.4, all models are wrong and all models agree on the same wrong class. This is the expected behavior when fixing seeds. The exception is Kaggle, as mentioned above. Not fixing seeds leads to a more even distribution of how many images are misclassified, see Fig. 3(b) and 4(b) (right).

Ranking the platforms: Fig. 5 shows the ranking of the platforms based on the classification error. The platform with the lowest error is ranked as 1. In case of ties, lower range in confidence interval wins. For TensorFlow 1.9, none of the platforms are consistently best for both CPU and GPU. Kaggle performs best and CodaLab performs worst on CPU, while the ranking is almost turned on the head for the GPU performance. This changes for TensorFlow 2.4, where Kaggle has the best result for CPU and for GPU when random seed is not fixed. When seeds are fixed on GPU, Kaggle is second best. Taking both versions of TensorFlow into account, the pattern is not very clear.

5.3. Discussion of empirical results

We were surprised to see that the results varied when conducting the experiment on CPU and fixing seed for the pseudorandom number generator, even though best practices were followed and the seed was fixed for the python environment, the built-in pseudo random generator in python, numpy, Keras and for the layers in the CNN that were configured with randomness, such as dropouts. Note that the training and test sets were kept the same for of all experiments. However, after investigating this surprising result, we found that the order that images were being fed to the models during training differed between epochs even though the random seeds were fixed. This was due to the fact that Keras library shuffled the training set by default, and it was not possible to configure it not to.[21] This is a bug that was fixed in the later version, as can be seen from the results produced by TensorFlow 2.4, although there still is a small variation in the results produced by Kaggle. Strangely, the bug seemed to be nonexistent in Codalab with TensorFlow 1.9. We were not surprised that the results vary on the GPU as it performs parallel processing that leads to some randomness [9].

Initializing the weights of a neural network can be viewed as purchasing a ticket in a lottery where one ticket is winning [36]. The winning ticket in the neural network initialization lottery is the initialization that leads to the best possible performance on the testset by the fully trained neural network. It follows from this that there are many non-winning tickets. However, in contrast to an ordinary lottery, the initialization lottery has different degrees of losers. There are tickets that lose the most, and these are the biggest losers. They have the worst possible performance on the testset. The rest of the tickets lie somewhere between the winning ticket and the biggest loser. We used the same seed for all experiments where seeds were fixed. For the TensorFlow 1.9 results, the fixed seed in our experiment might be closer to the biggest loser than the winning ticket. The mean classification error is higher when fixing the seed compared to not fixing it (same platform and same processing unit). For TensorFlow 2.4, it is the exact opposite. Frankle and Carbin [36] establish that it is architecture and data *in combination* that decides whether a seed has good or poor performance. Our results seem to suggest that ancillary software has a role to play as well, as the results changes between the two versions of TensorFlow (not processing unit as the results keep the same for GPU and CPU on the same version of TensorFlow). Our experiments are not able to decide whether the change is caused by the Keras bug or something else that has been changed between the two versions. For example, TensorFlow 2 changed from graph execution that was used in TensorFlow 1 to eager execution. Others have noted changes in results as well.[22]

5.4. Conclusion of the reproducibility experiment

The following conclusions are suggested by the results:

21 https://keras.io/api/models/model_training_apis

22 https://stackoverflow.com/questions/58441514/why-is-tensorflow-2-much-slower-than-tensorflow-1

1. **Platform:** Results can differ significantly between platforms even though the experiments are configured exactly the same in all other aspects. The results produced by Codalab(1.9,CPU,r) and CometML(1.9,CPU,r) is an example of this.
2. **Processing unit:** Results can differ significantly between processing units even though the experiments are configured exactly the same in all other aspects. The results produced Codalab(1.9,CPU,r) and Codalab(1.9,GPU,r) is an example of this.
3. **Software versions:** Results can differ significantly between different software versions even though the experiments are configured exactly the same in all other aspects. The results produced by CometML(1.9,CPU,r) and CometML (2.4,CPU,r) is an example of this.
4. **Software bugs:** A seemingly minor software bug can significantly change the results. This is exemplified by the average error between TensorFlow versions both for CPU and GPU when random seed is fixed (ALL in Table 6).
5. **Software complexity:** Fixing the random seeds to produce outcome reproducible results is not easy, as the random seed has to be set so many different places.
6. **Drawing conclusions:** When comparing the performance of two or more algorithms, the experiment is required to be reproduced in different laboratories for conclusive results. Ranking of the results shows that there is no machine learning platform that consistently performs best. However, as the different laboratories can produce seemingly statistically significant results without there being any difference in the experiment except for hardware and ancillary software, more laboratories are needed for conclusive results.

Gundersen [10] distinguishes between three degrees of reproducibility. We have shown that classifying images using a CNN on the same computer does not lead to the same outcomes and hence the results are not *outcome reproducible* for all experiments except CodaLab with fixed random seed on CPU. Furthermore, the results vary so much that false, but statistically significant conclusions can be drawn when performing the exact same analysis on the outcomes. Hence, the results are not *analysis reproducible*. We have not investigated inference reproducibility. Even when results are statistically significant, small effect sizes in machine learning experiments should lead to caution when concluding about which algorithms have the best performance. The experiments reported here support Ioannidis [37] who states that small effect sizes in the order of 1–1.5% often lead to false conclusions.

5.5. Evaluation

Are these conclusions valid? The main threat to the validity is that the experiment was only conducted five times for each of the 16 configurations on the two version of TensorFlow. A larger study with more runs per experiment configuration would increase the trust in the results. While more runs could change the mean and the range around the mean for the 95% confidence interval, it would not change the fact that changing hardware and ancillary software will result in different outcomes. Hence, results would still not be outcome reproducible. Also, the experiment is only conducted using a fairly simple convolutional neural network architecture. The conclusions are based on the assumption that when results vary with a fairly simple neural network architecture, they will also vary when the complexity of the architecture grows. This research does not provide information on whether increased complexity of the neural network architecture would lead to more or less variation. Finally, as we had no insights into the actual hardware and ancillary software being used to conduct the experiment, we cannot provided any information of how much one or the other affected the results.

6. Conclusion

First, we defined a framework for assessing how well machine learning platforms support reproducibility and used this framework to survey thirteen commercial and academic machine learning platforms. Then, we executed the exact same machine learning experiment on a selection of four of these machine learning platforms and compared the results. The results of our survey and reproducibility experiment illustrate at least some of the many barriers that make reproducibility so hard to achieve even when experiments are fully conducted on computers.

Even though there exist good solutions for covering each of the variables described by the framework for assessing the reproducibility of machine learning platforms, as described in Gundersen et al. [31], these solutions are not provided to users of the assessed machine learning platforms. Hence, the full *reproducibility potential* is not enabled for experiments conducted on the platforms we surveyed. Researchers have to take extra steps and put extra effort into making their research reproducible. Therefore, providing such features to the machine learning platforms is an important step for increasing the reproducibility of machine learning experiments. Although some of the platforms are better at supporting reproducibility, these are not the ones with the most users. Improving the reproducibility support on the platforms provided by the technology giants will have the highest impact on reproducibility of machine learning experiments. Providing these features will ease the documentation process when conducting machine learning experiments and enable provenance, transparency and sharing of the exact experiment conducted including textual documentation, code and data. However, contrary to popular belief, this will not ensure that independent investigators will be able to reproduce the results produced in the original experiment.

The reproducibility experiment conducted in the second part of this research paper illustrates this exact issue. The issue is that the actual hardware and ancillary software that are used to conduct the experiment affect the outcome of the experiment. This issue has received much less attention. In some cases, the difference in outcomes is so large that the wrong conclusion can be inferred based on the same analysis at the 95% confidence level. This is especially a problem with low effect sizes in the range of 1–1.5%, as the variation in outcomes caused by differences in hardware and ancillary software is on the lower side — at least in the experiment we conducted. Therefore, results with low effect sizes cannot be fully trusted unless the results are reproduced in many hardware and software configurations.

Concluding which machine learning algorithm performs better at a given task means selecting the best implementation of the algorithms and the best hyperparameter settings for each of them for the experiment. What our experiment shows is that the experiment is required to be reproduced in different laboratories for conclusive results, as different laboratories can produce seemingly statistically significant results without any difference in the experiment except for hardware and ancillary software. All comparative studies that are not reproduced in several laboratories must be considered exploratory if the effect sizes are small. In order to be considered confirmatory, studies with small effect sizes must be reproduced in several laboratories.

So, in conclusion, do machine learning platforms provide out-of-the-box reproducibility? The answer is no. More development is still needed, and in the end machine learning platforms can

only ensure a high reproducibility potential. Outcome reproducibility [10] is not necessarily guaranteed even when random seeds are fixed, as the hardware and ancillary software will vary over time. Hardware wears out and must be changed, and new versions of ancillary software are released and deployed. For some algorithms such changes will affect the outcomes produced when executing them. However, as long as the effect size is large enough, some variation in the outcomes will not change the conclusion.

CRediT authorship contribution statement

Odd Erik Gundersen: Conceptualization, Methodology, Writing – original draft, Validation, Visualization, Supervision. **Saeid Shamsaliei:** Software, Data curation, Visualization, Writing – original draft. **Richard Juul Isdahl:** Software, Data curation, Visualization, Writing – original draft.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

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

References

[1] M. Baker, 1,500 scientists lift the lid on reproducibility, Nature 533 (7604) (2016) 452–454, http://dx.doi.org/10.1038/533452a, http://dx.doi.org/10.1038/533452a.

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

[3] B.A. Nosek, G. Alter, G.C. Banks, D. Borsboom, S.D. Bowman, S.J. Breckler, S. Buck, C.D. Chambers, G. Chin, G. Christensen, et al., Promoting an open research culture, Science 348 (6242) (2015) 1422–1425.

[4] M.L. Braun, C.S. Ong, Open science in machine learning, Implementing Reproducible Research (2014) 343–345.

[5] C. Collberg, T.A. Proebsting, Repeatability in computer systems research, Commun. ACM 59 (3) (2016) 62–69, http://dx.doi.org/10.1145/2812803, http://dx.doi.org/10.1145/2812803.

[6] A. Torralba, A.A. Efros, Unbiased look at dataset bias, in: CVPR 2011, IEEE, 2011, pp. 1521–1528.

[7] P. Henderson, R. Islam, P. Bachman, J. Pineau, D. Precup, D. Meger, Deep reinforcement learning that matters, 2018, https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/16669.

[8] S.-Y. Hong, M.-S. Koo, J. Jang, J.-E.E. Kim, H. Park, M.-S. Joh, J.-H. Kang, T.-J. Oh, An evaluation of the software system dependency of a global atmospheric model, Mon. Weather Rev. 141 (11) (2013) 4165–4172, http://dx.doi.org/10.1175/MWR-D-12-00352.1.

[9] P. Nagarajan, G. Warnell, P. Stone, Deterministic implementations for reproducibility in deep reinforcement learning, in: AAAI 2019 Workshop on Reproducible AI, 2019.

[10] O.E. Gundersen, The fundamental principles of reproducibility, Phil. Trans. R. Soc. A Volume 379 (297) (2021).

[11] C. Drummond, Replicability is not reproducibility: nor is it good science, in: ICML Workshop, 2009.

[12] V.C. Stodden, Trust your science? Open your data and code, Amstat. News (2011) 21–22.

[13] R.D. Peng, Reproducible research in computational science, Science 334 (6060) (2011) 1226–1227.

[14] S.N. Goodman, D. Fanelli, J.P.A. Ioannidis, What does research reproducibility mean? Sci. Transl. Med. 8 (341) (2016) http://dx.doi.org/10.1126/scitranslmed.aaf5027, 341ps12–341ps12.

[15] O.E. Gundersen, S. Kjensmo, State of the art: Reproducibility in artificial intelligence, in: Thirty-Second AAAI Conference on Artificial Intelligence, 2018.

[16] R. Tatman, J. VanderPlas, S. Dane, A practical taxonomy of reproducibility for machine learning research, 2018.

[17] E. National Academies of Sciences, Medicine, et al., Reproducibility and Replicability in Science, National Academies Press, 2019.

[18] B. Devezer, L.G. Nardin, B. Baumgaertner, E.O. Buzbas, Scientific discovery in a model-centric framework: Reproducibility, innovation, and epistemic diversity, PLOS ONE 14 (5) (2019) 1–23, http://dx.doi.org/10.1371/journal.pone.0216125.

[19] H.E. Plesser, Reproducibility vs. Replicability: A brief history of a confused terminology, Front. Neuroinf. 11 (2018) 76, http://dx.doi.org/10.3389/fninf.2017.00076.

[20] J.B. Buckheit, D.L. Donoho, Wavelab and reproducible research, Standford, CA, 1995, http://statweb.stanford.edu/~wavelab/Wavelab_850/wavelab.pdf.

[21] J.F. Claerbout, M. 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. http://sepwww.stanford.edu/doku.php?id=sep:research:reproducible:seg92.

[22] Y. Gil, C.H. David, I. Demir, B.T. Essawy, R.W. Fulweiler, J.L. Goodall, L. Karlstrom, H. Lee, H.J. Mills, J.-H. Oh, et al., Toward the geoscience paper of the future: Best practices for documenting and sharing research from data to software to provenance, Earth Space Sci. 3 (10) (2016) 388–415.

[23] A. Sethi, A. Sankaran, N. Panwar, S. Khare, S. Mani, DLPaper2Code: Auto-generation of code from deep learning research papers, in: Thirty-Second AAAI Conference on Artificial Intelligence, 2018.

[24] J. Lasser, Creating an executable paper is a journey through Open Science, Commun. Phys. 3 (1) (2020) 1–5.

[25] J. Grus, I do not like notebooks, 2018, Talk at JupyterCon 2018.

[26] J.M. Perkel, Reactive, reproducible, collaborative: computational notebooks evolve, Nature 593 (7857) (2021) 156–157.

[27] M.D. Wilkinson, M. Dumontier, I.J. Aalbersberg, G. Appleton, M. Axton, A. Baak, N. Blomberg, J.-W. Boiten, L.B. da Silva Santos, P.E. Bourne, et al., The FAIR guiding principles for scientific data management and stewardship, Sci. Data 3 (2016).

[28] V. Stodden, M. McNutt, D.H. Bailey, E. Deelman, Y. Gil, B. Hanson, M.A. Heroux, J.P. Ioannidis, M. Taufer, Enhancing reproducibility for computational methods, Science 354 (6317) (2016) 1240–1241.

[29] J. Starr, E. Castro, M. Crosas, M. Dumontier, R.R. Downs, R. Duerr, L.L. Haak, M. Haendel, I. Herman, S. Hodson, et al., Achieving human and machine accessibility of cited data in scholarly publications, PeerJ Comput. Sci. 1 (2015) e1.

[30] A. Cockburn, P. Dragicevic, L. Besançon, C. Gutwin, Threats of a replication crisis in empirical computer science, Commun. ACM 63 (8) (2020) 70–79.

[31] O.E. Gundersen, Y. Gil, D. Aha, Towards reproducible research, open science, and digital scholarship in AI publications, AI Mag. 39 (3) (2018) 56–68, http://dx.doi.org/10.1609/aimag.v39i3.2816, https://www.aaai.org/ojs/index.php/aimagazine/article/view/2816.

[32] A. Anjos, L. El Shafey, S. Marcel, BEAT: An open-science web platform, in: Thirty-Fourth International Conference on Machine Learning, 2017, https://openreview.net/group?id=ICML.cc/2017/RML. URL https://beat-eu.org/platform.

[33] D. Sculley, G. Holt, D. Golovin, E. Davydov, T. Phillips, D. Ebner, V. Chaudhary, M. Young, Machine learning: The high interest credit card of technical debt, 2014.

[34] F. Chollet, et al., Keras: The python deep learning library, Astrophys. Source Code Libr. (2018) ascl–1806.

[35] O.E. Gundersen, S. Shamsaliei, R.J. Isdahl, Do machine learning platforms provide out-of-the-box Reproducibility? (experiments), 2021, http://dx.doi.org/10.5281/zenodo.4661935.

[36] J. Frankle, M. Carbin, The lottery ticket hypothesis: Finding sparse, trainable neural networks, in: International Conference on Learning Representations, 2018.

[37] J.P.A. Ioannidis, Why most published research findings are false, PLOS Med. 2 (8) (2005) http://dx.doi.org/10.1371/journal.pmed.0020124.

Odd Erik Gundersen is an adjunct associate professor at the Norwegian University of Science and Technology and the Chief AI Officer at TrønderEnergi, a Norwegian renewable energy company. Gundersen has applied AI in industry, mostly for startups, since 2006. Currently, he is investigating how AI can be applied in the renewable energy sector and for driver training as well as how research in AI can be made reproducible.

Saeid Shamaliei is a PhD student at the Norwegian University of Science and Technology with a specialization in artificial intelligence. Mr. Shamaliei's research interests are deep learning, machine learning and reproducibility.

Richard Juul Isdahl received his master's degree in computer science with a specialization in artificial intelligence from the Norwegian University of Science and Technology in 2019. His research interest are machine learning and reproducibility.