A Practical Taxonomy of Reproducibility for Machine Learning Research

Rachael Tatman
Kaggle
Seattle, WA 91803
rachael@kaggle.com

Jake VanderPlas
eScience Institute
University of Washington
Seattle, WA 98195
jakevdp@uw.edu

Sohier Dane
Kaggle
Seattle, WA 91803
sohier@kaggle.com

Abstract

Discussions of reproducibility in science are often framed from the perspective of scientists and researchers who want to validate published claims. A complementary perspective is that of the practitioner who sets out to apply a new computational method within their own domain, the first step of which is often to reproduce the published results as a check for correctness of code. In this paper we discuss a taxonomy of reproducibility from this perspective of a practitioner. *Low reproducibility* studies are those which merely describe algorithms, *medium reproducibility* studies are those which provide the code and data but not the computational environment in which the code can be run, and *high reproducibility* studies are those which provide the code, data, and full computational environment necessary to reproduce the results of the study. We identify some exemplars of each of these types of reproducibility from the machine learning literature, motivate the case for high reproducibility studies, and discuss concrete tools and strategies for researchers who wish to ensure easy adoption of their methods by practitioners.

1 Introduction

The ever increasing size of datasets and availability of computational resources in recent years has proven both a boon and a challenge to scientific researchers. On one hand, the large datasets and complex models enable discovery of new and more subtle scientific phenomena; on the other hand, the data-intensive and computationally-intensive methods of modern research has begun to outgrow older methods for communicating research results. In particular, the scientific paper, which was classically designed to include everything necessary to replicate any particular study, is no longer a suitable format to describe complex methods in enough detail that they can be exactly reproduced.

In this setting, scientists from a variety of fields have begun discussing the need for research to be reproducible, that is, to make available the code and data required for a reader to recreate research results [1, 2, 3, 4]. These discussions often focus more on reproducible research from the standpoint of scientific purity; take for example the oft-repeated quotation from Buckheit & Donoho [5] paraphrasing Jon Claerbout [6]:

> An article about computational results is advertising, not scholarship. The actual scholarship is the full software environment, code and data, that produced the result.

From the standpoint of a machine learning (ML) practitioner, such lofty ideals are appreciated, but often far separated from the day-to-day task of implementing and applying novel ML methods. In this case the reproducibility question becomes one of pragmatism: what concrete artifacts should

2nd Reproducibility in Machine Learning Workshop at ICML 2018, Stockholm, Sweden.

researchers include with their publications to allow practitioners to understand the methods and apply them in their own domains?

This paper aims to provide resources to answer this question, and discuss practical aspects of addressing the reproducibility problem within the context of the current ecosystem of open tools.

1.1 What is reproducibility and why do we care?

We use "reproducibility" to mean recreating the exact results reported by the author. We are not considering the case where someone wants to apply a specific technique to a new dataset. For clarity, we will call the former "reproduction" and the latter "replication", after Claerbout and Karrenbach [6] but acknowledge that these terms are not always used in this way [7] (see [8] for a detailed discussion).

In the body of this paper, we lay out an updated version of the Reproducibility Spectrum proposed by Peng [1] that more closely aligns with current practices in sharing code and data. Our version has three levels of reproducibility: low, medium, and high. (These are equivalent the the lowest, middle and highest levels of reproducibility as discussed by Peng.) For each level, we provide an example ICML paper along with concrete suggestions for how papers at this level of reproducibility can be made more reproducible. Finally, we discuss some considerations for long-term reproducibility.

Our interest in reproducibility is a practical one. We have found that non-academic machine learning practitioners often consider reproducing an existing study the first step in applying the methods to a new dataset or domain. Checking the output of their code against the results of the paper provides instant feedback that the code was run correctly. A reproducible machine learning study can be thought of as the "sample code" that allows others to apply researchers' contributions and, as a result, ensuring reproducibility increases the broader impact of machine learning research.

2 Low reproducibility: Finished paper only

A well-written paper should, at least in theory, explicitly lay out the researchers methodology in enough detail that another expert in the field should be able to reproduce it. However, "from scratch" reproducibility is often impractical or impossible given time constraints and missing or incomplete information.

An example of a paper with a software contribution at this level of reproducibility is "Combining Online and Offline Knowledge in UCT" by Sylvain Gelly and David Silver [9], which was initially published in 2007 and was awarded the ICML "Test of Time Award" in 2017. It is perhaps unsurprising that this paper was not distributed with its code and data: GitHub, the platform most commonly used to share code at the time of this writing, wasn't founded until a year after the paper was published. However, given the current ease of sharing code and data online, we recommend against releasing papers with a code contribution at this level of reproducibility.

3 Medium reproducibility: Code and data

The next level of reproducibility is to provide both the data and code, with the former anonymized as necessary. (For ML specifically, we don't believe it makes sense to distribute only the code. For approaches like reinforcement learning where data is generated during learning, the generated data should also be shared.) This is the level of reproducibility encouraged by ICML. From the 2018 ICML style guidelines as most recently updated by Iain Murray: "We strongly encourage the publication of software and data with the camera-ready version of the paper whenever appropriate. This can be done by including a URL in the camera-ready copy."

An example of an ICML paper at this level of reproducibility is the paper "Lost Relatives of the Gumbel Trick" [10] which was awarded an Honorable Mention in 2017. The code and data were both provided in a GitHub repository[1]. The materials provided by the authors are sufficient to reproduce the findings and figures presented in the paper.

[1] https://github.com/matejbalog/gumbel-relatives

3.1 Improving reproducibility at this level:

A medium-reproducibility study can require significantly more investment in time, on the part of the researcher, when compared to a low-reproducibility study. We recommend the following practical steps:

- Make your project as modular as possible: a reproducer should ideally be able to process data, train the model or evaluate the trained model without needing to complete any of the other steps.
- Document the original environment used to run the code, including all dependencies with their versions and subversions, and distribute this documentation along with your code and data. This may be done using lockfiles, requirement files, config files or similar tools.
- Ensure that your code and data are licensed for reuse and distribute the license with your project. The MIT license is a permissive open source license and is commonly used for research code, but see Morin et al. [11].
- Before sharing your code, reproduce your project yourself on a new machine (if possible). Note what problems you run into and either modify your project to avoid them or address them in your documentation.

4 High reproducibility: Code, data and environment

The highest level of reproducibility includes sharing not only the code and data, but also the environment used to run that code. Peng refers to this level of reproducibility as "linked and executable code and data" [1]. By "environment" here, we mean all the libraries and dependencies necessary to run the code provided on a new machine. Providing the environment is a step above the level of reproducibility suggested by others [12]. It is, however, necessary in order to avoid the "it runs on my machine" problem, where a project can only be reproduced by running it on the same computer it was originally written on.

There are currently three main techniques for sharing a computational environment. In order of increasing time investment on the researcher's part, they are: using a hosting service, providing a container, and providing a virtual machine image. Below we discuss the benefits and drawbacks of each.

The ICML 2017 best paper, "Understanding Black-box Predictions via Influence Functions" by Koh and Laing [13], employed two of these approaches: using a hosting service to share their code and data so that reproducers can execute their code directly from a web browser[2] and also creating and sharing a Dockerfile [3]. This represented an exceptionally high level of reproducibility, and we applaud the authors' efforts.

4.1 Hosting services

The simplest solution to providing the code, data and environment is to distribute research results using a hosted service that will allow reproducers to execute the code in their browsers. This is faster for both researchers (as the logistics of hosting and distributing code is done by the host) and for reproducers (as they do not need to set up their own environment).

Many services exist for hosting data and executable code. These include for-profit services such as Kaggle Kernels, Google Colaboratory, Amazon SageMaker, IBM Watson Studio, and Microsoft Azure Notebooks (most of which offer a limited free tier); not-for-profit services such as MyBinder, PanGeo, and Codalab; as well as a variety of similar services offered by smaller startup companies.

The main drawback of hosted services is that most have limits on the size of data and the amount of compute available. For example, as of this writing both Kaggle Kernels and Google Colaboratory provide free use of notebooks running on NVIDIA Testla K80 GPUs, but limit run-times to six hours and twelve hours, respectively. While this may be useful for many, it is not sufficient for all research projects.

[2] `https://worksheets.codalab.org/worksheets/0x2b314dc3536b482dbba02783a24719fd/`

[3] `https://hub.docker.com/r/pangwei/tf1.1/`

Describing the rapidly-developing landscape of hosted services and their strengths and weaknesses is well beyond the scope of this paper, and we encourage researchers to compare various options and choose the one that is the best fit for their needs.

4.2 Containers

Containers allow researchers to combine code, data and all the dependencies needed to run the code in a single portable format. Code executed inside a container can only access and interact with the other contents of the container, which ensures that all the dependencies necessary to run code have been included. The biggest drawback of containers is that they use the local operating system of the machine they are run on. Thus, a container made on a computer running Linux will not run on a computer running Windows without being run from within a virtual machine. Docker is currently the most popular container format, but standards set by Open Container Initiative should ensure some level of portability between formats.

4.3 Virtual machines (VMs)

Virtual machines are similar to containers, but also include the entire operating system used to run the code. As a result, they can be more portable across different machines. However, they are generally orders of magnitude larger and can require much longer startup times. VirtualBox is one popular open source option for creating and sharing virtual machines.

Regardless of how the computational environment is shared, there are some general guidelines that can help improve the ease of reproduction of projects using this framework.

4.4 Improving reproducibility at this level:

A high-reproducibility study can require substantial effort on the researcher's part. In addition to the above resources, we recommend the following practical steps:

- Try to minimize the number of steps reproducers will need to perform. For example, move as much code as possible into scripts that can be batch called from a notebook or include a single script that acquires the data, prepares the environment, and executes the code with a single command.
- If not using a hosted service, include instructions on how to download and set up the docker or VM. You can do this by setting up the project from scratch on a new computer (ideally with a different operating system) and writing down each step as you do it.

5 Discussion & Conclusion

Issues of reproducibility in ML extend beyond ICML. Of the 679 papers presented at NIPS in 2017, for instance, only 259–or less than 40%–provided links to code on the NIPS website. Far fewer provided the environment needed to actually run that code. A few notable exceptions include Liu et al.'s paper on unsupervised image-to-image translation networks [14] and the papers presented at the MLTrain NIPS workshop. As a field, ML is making strides towards reproducibility, but there is still a long way to go.

There are other challenges in reproducibility beyond encouraging researchers to adopt reproducible workflows. In particular, it's important to consider the longevity of reproducible examples. For instance, if a graduate student is hosting code on their university's website, will it remain available after they graduate? These concerns are not just hypothetical: the rate of decay in links in published papers is very high. One study of NLP papers found that roughly 20% of links were broken or deprecated within five years of publication [15]. Until the field converges on a long-term archival solution, the best option appears to be for researchers to maintain reproducible examples in institution-independent platforms and to share projects redundantly across multiple platforms.

Regardless of what changes the future brings to the ML research landscape, the relative reproducibility of the approaches discussed here should remain stable. Sharing code and data is an important first step to improving reproducibility, but sharing the computational environment increases the ease and longevity of reproducible research projects.

References

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

[2] Victoria Stodden, Jonathan Borwein, and David H Bailey. Setting the default to reproducible in computational science research. *SIAM News*, 46(5):4–6, 2013.

[3] 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.

[4] Justin Kitzes, Daniel Turek, and Fatma Deniz. *The practice of reproducible research: case studies and lessons from the data-intensive sciences*. Univ of California Press, 2017.

[5] Jonathan B Buckheit and David L Donoho. Wavelab and reproducible research. In *Wavelets and statistics*, pages 55–81. Springer, 1995.

[6] Jon F Claerbout, Martin Karrenbach, et al. Electronic documents give reproducible research a new meaning. In *1992 SEG Annual Meeting*. Society of Exploration Geophysicists, 1992.

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

[8] B Marwick, A Rokem, and V Staneva. Assessing reproducibility, 2017.

[9] Sylvain Gelly and David Silver. Combining online and offline knowledge in UCT. In *Proceedings of the 24th international conference on Machine learning*, pages 273–280. ACM, 2007.

[10] Matej Balog, Nilesh Tripuraneni, Zoubin Ghahramani, and Adrian Weller. Lost relatives of the Gumbel trick. In *34th International Conference on Machine Learning (ICML)*, August 2017.

[11] Andrew Morin, Jennifer Urban, and Piotr Sliz. A quick guide to software licensing for the scientist-programmer. *PLoS computational biology*, 8(7):e1002598, 2012.

[12] Roger D Peng, Francesca Dominici, and Scott L Zeger. Reproducible epidemiologic research. *American journal of epidemiology*, 163(9):783–789, 2006.

[13] Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions. In *International Conference on Machine Learning*, pages 1885–1894, 2017.

[14] Ming-Yu Liu, Thomas Breuel, and Jan Kautz. Unsupervised image-to-image translation networks. In *Advances in Neural Information Processing Systems*, pages 700–708, 2017.

[15] Margot Mieskes. A quantitative study of data in the NLP community. In *Proceedings of the First ACL Workshop on Ethics in Natural Language Processing*, pages 23–29, 2017.