Tools for Reproducible Research in Machine Learning

Nicolai Anton Lynnerup[*†], John Hallam[†] and Rasmus Hasle[*]
[*]Robot Technology, Danish Technological Institute (DTI)
[†]Embodied Systems for Robot Learning, University of Southern Denmark (SDU)
Email: [*]{nily, raha}@dti.dk, [†]{nia, john}@mmmi.sdu.dk

***Abstract*—To make reproducible research more tractable, intelligent and easy-to-use tools must be presented to a broad audience of researchers. Through examples in both reinforcement learning (RL) and deep learning (DL) we demonstrate how a chosen series of tools can be efficiently used for creating reproducible research within computational sciences. We show that using a reasonably standardised folder structure, a common way to describe experiments (scientific trials) and a visual experiment tracker can assist others (and future-you) in understanding and learning from the conducted trials.**

I. Motivation

Why should I care? Others (and future-you) will thank you! Following a standardised folder structure will ease the overview of the project and help others collaborate more easily with you. It further allows other researchers to learn from your analysis and feel confident in the conclusions at which the analysis arrives. In the end you are actually helping yourself. Your closest collaborator is future-you, and future-you does not reply to e-mails! You might think that remembering what you did will be good enough, but all the small details will vanish eventually.

We acknowledge that doing things right, takes more time in the short-term, but it is an investment to save time and aggravation in the long-term.

II. Considerations

Use a consistent folder structure, but also know when to be inconsistent! The tools you use, should ease the process of creating, running and reporting scientific experiments, not hinder it. For the examples provided through this work, we showcase the Cookiecutter Data Science approach, from which the following considerations are inspired.

Use a naming convention for files, folders and code. We will conform to the Smithsonian Data Management Best Practices[1] which describes five precepts of file naming and organisation. We attempt to make the principally clear definitions of the precepts less operationally elusive by example[2]. Also here, know when to be inconsistent.

The work presented in this paper is partially funded by the Danish Technological Institute and partially Innovation Fund Denmark through their Industrial Researcher Program, grant 8053-00057B.

[1]https://library.si.edu/sites/default/files/tutorial/pdf/filenamingorganizing20180227.pdf

[2]Also see https://style.tidyverse.org/

Use open data formats whenever possible as it means that your code and results can be used by all researchers, not only those who can afford the license.

Consider data as immutable. Never do any processing on the raw data or its meta-data, especially not manual edits. Don't do inline modifications to your raw data. The code you write should move the raw data through a pipeline, creating new data files or results as separate outputs. As data is immutable it needs no version control, so put the data folder in the `.gitignore` file, or equivalent. Host the data on other, more suitable platforms, such as; AWS S3, Git large file storage, or your own server.

Analysis is a directed acyclic graph (DAG). For easing the complete analysis, split (long-running) experiments into parts, removing the need to rerun the complete analysis-process exhaustively every time. Open source tools such as Make[3] can assist managing steps which depend on one-another.

Seed the stochastic processes. Machine learning (ML) is permeated with stochastic processes why seeding the random number generators (with a preset number of seeds) is a strict requirement for obtaining reproducibility. Also, remember to report the seeds used for obtaining different results.

Collect all hyper-parameters in a single configuration file or use an experiment tracker in which all parameters are recorded.

Share the environment. One of the largest issues of reproducing an analysis is always how to reproduce the computational environment. For Python, the package manager `pip`[4] could be used to generate a list of required software dependencies (`pip freeze > requirements.txt`) or for analysis which requires specific underlying operating systems Docker containers could be beneficial. In addition to letting others avoid the *dependency hell*, future-you (who wants/might need to rerun the analysis) will also benefit from wrapping your analysis into a docker container.

Do (intelligent) experiment tracking. Find or develop a system for intelligent experiment tracking. Prefer one that can link with git (or similar version control) such that a git

[3]https://www.gnu.org/software/make/

[4]https://pypi.org/project/pip/

hash is stored with each trial conducted. For this work we extend the open source platform Guild AI[5], such that each trial is linked to a git commit. Guild AI were chosen as it is open source, has good community support and provides a web-based solution for visualisation of results in addition to the command line interface (CLI).

Write meta-data for describing how the workflow is. What commands are intended to be called in what sequence and with what arguments.

Keep secrets out of version control. Don't put your private key or password in files going to git. Instead create an environment file (`.env`) with all keys and blacklist that file from version control. Remember to create an example file so the reader will know how to construct his/her own.

Be conservative in changing the default folder structure across all projects but be liberal for those (few) projects that truly requires another structure.

III. Methodology

We combine the aforementioned tools in a Python-based CLI with Bash auto-completion[6], making it easier to overview the underlying functionality while maintaining cross-platform abilities of underlying frameworks. Our goal is to keep it as well-structured, flexible and lightweight as possible for making the lives of our (future) collaborators easier.

Our tool, Tracker[7], is an attempt at combining the efforts of previous work, creating one uniform work-flow. Tracker will assist at every stage from creating the project and its very first files, to reporting the final results.

Validation of Work-flow: We will validate our developed tool through several examples within both supervised deep learning (DL) and reinforcement learning (RL).

IV. Discussion

Should we all conform to one common tool for reproducibility? No we don't believe that to solve anything. Choosing the right tools is a subjective task and many has many opinions. Through this work we illustrate the use of a selected few amongst many.

Trade-off between velocity of research and a rigorous analysis. Often, arguments against reproducible research contains the excuse that research must evolve rapidly "to keep up". Contrary, one could argue that irreproducible research is not actually research, merely data – and poor quality of data at that – and that there is not really a trade-off between going somewhere carefully and possibly nowhere fast [1].

[5]https://guild.ai/

[6]Using the Python package, click: https://click.palletsprojects.com/

[7]https://github.com/dti-research/tracker

References

[1] Nicolai Anton Lynnerup et al. "A Survey on Reproducibility by Evaluating Deep Reinforcement Learning Algorithms on Real-World Robots". In: *Proceedings of the 3rd International Conference on Robot Learning - Volume 100*. (Osaka, Japan). Proceedings of Machine Learning Research (PMLR), 2019.