Wiresharks: Reproducibility Project for HACKHPC@ADMI25

Project Overview

Our core objective for the ADMI25 Hackathon is to delve into the critical area of reproducible science within computing research. As Team Wiresharks, we are undertaking the hands-on task of reproducing existing scientific projects hosted on GitHub. Our primary aim is to thoroughly assess how straightforward or challenging it is for others to replicate previously published research. This document outlines our goals, team structure, process, and findings in reproducible science.

Goals:

Our main goal is to evaluate how easy it is to reproduce existing papers with code repositories. We do this by answering key questions:

- Did the repo include all necessary code?
- Was the dataset public and accessible?
- Were instructions up to date?
- Any issues with dependencies or hardware?
- How close were your results to theirs?

By documenting every step and challenge, we assign a reproducibility rating, offering insights into practical scientific replication.

How We Address Reproducible Science

Our project directly addresses reproducible science by:

- Attempting to Replicate: We try to rebuild and run existing research projects.
- Evaluating Challenges: We identify common roadblocks like outdated instructions, Python version issues, and missing datasets.
- Providing Feedback: Our detailed reports and ratings help the scientific community understand what makes a project truly reproducible.

How We Evaluated HPC Papers

We picked papers with code from areas like HPC or data science (e.g., from ICSE conferences). We used a detailed checklist to score them on:

- Paper Availability: Can you get the paper (e.g., free online)?
- Code & Software: Is the code on GitHub with setup steps?
- Datasets: Is the data available and described?
- Computer Needs: What hardware (CPU, memory) is required?
- GPU Needs: Are special GPUs needed, and what kind?
- Documentation Quality: Are the instructions clear and helpful?
- Ease of Setup: How simple is it to get the project running?
- Reproducibility of Results: Can we get the same answers as the paper?

Project 1: Business Flow Tampering (BFT) Detector

What it does: Automatically finds ways to bypass website paywalls and ads.
Our Experience:

- Code was available, but instructions were very outdated.
- Major Python version and software conflicts prevented it from running.
- Hardware issues (Docker/virtualization) were a dead end.
- Couldn't run the project to compare our results.
- Reproducibility Rating: 2/5 (Very Difficult)

Project 2: Fairify

What it does: Formally checks AI models for fairness (ensures similar people get similar outcomes).
Our Experience:

- Paper, Code, and Data were all available.
- Python version conflicts (needed older TensorFlow) required complex setup.
- Critical Problem: Script ran for hours, but NEVER produced any output files.
- We could run the project, but without results, we couldn't verify its claims.
- Reproducibility Rating: 3/5 (Doable)

Project 3: Bad Snakes

What it does: Evaluates tools designed to find malicious Python packages on PyPI.
Our Experience:

- Paper and Code were open-access and easily found.
- Python version and dependency conflicts also caused setup issues.
- Major Problem: The actual original datasets used in the paper were NOT available due to privacy.
- Missing crucial data prevented us from verifying their quantitative results.
- Reproducibility Rating: 2/5 (Very Difficult)

Project	Paper Avail.	Code & Software	Datasets Avail.	Computer Reqs.	GPU Reqs.	Doc. Quality	Ease of Setup	Reprod. of Results	Overall Rating
BFT Detector	3	3	2	3	1	2	2	1	2 (Very Difficult)
Fairify	4	4	5	3	5	2	2	1	3 (Doable)
Bad Snakes	5	5	1	2	1	2	2	1	2 (Very Difficult)

Score	Paper Availability	Availability of Code and Software	Availability of Datasets	Computer Requirements
1	Unavailable/Impossible to find.	No code/Private repository.	Dataset missing.	No info/Impossible to meet.
2	Paywalled/Very hard to access.	Code available, but major parts missing/broken.	Mentioned, but completely inaccessible.	Vague/Requires rare hardware.
3	Available, but via obscure link.	Code available, incomplete/needs big fixes.	Available, but very hard to find/access.	Specific, but hard to meet.
4	Open-access, direct link.	Code available, mostly complete, minor issues.	Available, but metadata is poor/incomplete	Specific, but common hardware.
5	Open-access, easily searchable.	Code fully available, complete.	Fully accessible with complete metadata.	Clear, common, and flexible hardware.
	GPU Requirements	Documentation Quality	Ease of Setup	Reproducibility of Results
1	No info/Mandatory custom GPU.	None or misleading.	Cannot be run due to critical issues or missing parts.	Cannot be run due to critical issues or missing parts.
2	GPU required, vague/high-end sp	Very poor	Can't run without major problems; needs expert help or significant workarounds.	Can't run without major problems; needs expert help or significant workarounds.
3	GPU optional, specific specs.	Basic or needs much interpretation.	Can be run with some effort; requires troubleshooting or minor fixes.	Can be run with some effort; requires troubleshooting or minor fixes.
4	GPU optional, common specs.	Clear but few details.	Runs well with minimal effort; minor adjustments might be needed.	Runs well with minimal effort; minor adjustments might be needed.
5	No GPU required (CPU-only).	Comprehensive & clear.	Runs perfectly by simply following the instructions; no issues.	Runs perfectly by simply following the instructions; no issues.

References

- S. Biswas and H. Rajan, "Fairify: Fairness Verification of Neural Networks," 2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE), Melbourne, Australia, 2023, pp. 1546-1558, doi: 10.1109/ICSE48619.2023.00134. keywords: {Computational modeling;Scalability;Neurons;Artificial neural networks;Production;Machine learning;Biological neural networks;fairness;verification;machine learning},
- D. -L. Vu, Z. Newman and J. S. Meyers, "Bad Snakes: Understanding and Improving Python Package Index Malware Scanning," 2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE), Melbourne, Australia, 2023, pp. 499-511, doi: 10.1109/ICSE48619.2023.00052. keywords: {Ecosystems;Malware;Security;Time factors;Indexes;Interviews;Open source software;Open-source software (OSS) Supply Chain;Malware Detection;PyPI;Qualitative Study;Quantitative Study},
- I. L. Kim, W. Wang, Y. Kwon and X. Zhang, "BFTDETECTOR: Automatic Detection of Business Flow Tampering for Digital Content Service," 2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE), Melbourne, Australia, 2023, pp. 448-459, doi: 10.1109/ICSE48619.2023.00048. keywords: {Fault diagnosis;Analytical models;Web services;Software algorithms;Motion pictures;Business;Software engineering;JavaScript;business flow tampering;dynamic analysis;vulnerability detection}
- Gemini: Web scraping, grammar, and code debuggin

SGX3 Extend. Expand. Exemplify.

ADMI

Technologies Used

- Programming Languages: Python, JavaScript
- Core Libraries/Frameworks: TensorFlow/Keras, Numpy, Z3, Poetry, Bandit, scikit-image
- Automation/Virtualization: Google Puppeteer, Docker (attempted), pyenv
- Environment: Kali Linux (WSL)

Common Challenges

- Outdated Instructions: Frequent source of setup failures.
- Python Versioning: Newer Python often breaks older project dependencies.
- Missing Datasets: A significant barrier to full reproduction.
- Environment Specificity: Unexpected hardware or system settings blocking progress.
- Lack of Clear Output: Scripts running but no results appearing.

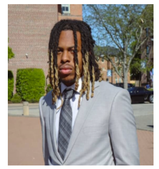
What We Learned (Key Takeaways)

Making science reproducible is hard! For projects to be easier to reproduce, they need:

- Clear, up-to-date instructions.
- Easy-to-get data (or clear alternatives).
- Simple setup steps (e.g., using tools like Docker).
- No assumptions about what computer users have.

Meet The Sharks

Ayinde Hooks



Ryan Grimes

Howard “Shiloh” Ames

Auiana D’Avilar