

Supplementary Materials

When code does not run: reproducibility challenges in materials machine learning benchmarks

Bohui Lyu¹, John Bonini², Mao Zhang¹, Amin Sadeghi³, Ali Jaber³, Jason Hattrick-Simpers^{3,4}, Kamal Choudhary^{2,5}, Daniel Wines², Kangming Li¹

¹Division of Physical Science and Engineering, King Abdullah University of Science and Technology (KAUST), Thuwal 23955-6900, Saudi Arabia.

²Material Measurement Laboratory, National Institute of Standards and Technology, Gaithersburg, MD 20899, USA.

³Department of Materials Science and Engineering, University of Toronto, Toronto M5S 3E4, Canada.

⁴Acceleration Consortium, University of Toronto, Toronto M5S 3E4, Canada.

⁵Department of Electrical and Computer Engineering, Whiting School of Engineering, Johns Hopkins University, Baltimore, MD 21218, USA.

Correspondence to: Kangming Li, Division of Physical Science and Engineering, King Abdullah University of Science and Technology (KAUST), Thuwal 23955-6900, Saudi Arabia. E-mail: kangming.li@kaust.edu.sa

Supplementary Method 1. Pseudocode for the automated, error-guided remediation workflow (Stage 2)

Algorithm 1: Auto-fix environment from error logs

Input:

submission metadata

submission source files

Python version (3.6-3.12)

maximum attempts = 5

1. Create a clean Conda environment with the specified Python version.
2. Extract and deduplicate dependency strings from requirements.python.
3. For attempt = 1, ..., 5:

 Run pip install using the current dependency list.

 If installation succeeds:

 proceed to import validation.

 Else if pip reports an unavailable requirement:

 if the requirement contains a local build suffix:

 remove the local build suffix.

 else:

 remove the version pin from the implicated package.

 Else if pip reports conflicting dependencies:

 extract package names from the conflict summary.

 remove version pins only from those packages.

 Else:

 remove all version constraints as a final fallback.

4. If installation does not succeed after five attempts:

 classify Stage 2 as failed.

5. Install MatBench without a version constraint if it is absent.

6. Convert root-level notebooks to Python scripts.

7. Extract statically declared top-level absolute imports.

8. Import all detected modules.

9. If every import succeeds:

 classify Stage 2 as successful;

 export the reconstructed environment.

Else:

 classify Stage 2 as failed.

Supplementary Method 2. LLM-assisted troubleshooting

During Stage 3, GPT-5.4 Thinking (OpenAI; released March 5, 2026) was accessed through the ChatGPT web interface. No API was used, and no API-level parameters, such as temperature, were configured or recorded. For unresolved environments, we provided the model with the required package specifications, attempted installation commands, Python version, and corresponding installation or import error messages. The model was asked to identify possible causes and suggest troubleshooting steps, including alternative package versions, changes in installation order, or combined pip and conda installation strategies. No fixed prompt template or predefined maximum number of dialogue rounds was used. The model had no direct access to the terminal or repositories and did not execute commands or modify files. All suggestions were manually reviewed and tested by the authors, and a proposed change was retained only if the environment could be installed successfully and passed the predefined import

check. A maximum of ten dialogue rounds was permitted for each submission. If the environment could not be installed successfully or failed the predefined import check after ten rounds, the environment reconstruction was classified as unsuccessful.