# REPA: Reproducibility Evaluation via an Autonomous Pipeline Architecture

**Nura Tamton** [1] **Yisong Miao** [1] **Min-Yen Kan** [1]

## Abstract

We present **REPA**, a framework that reproduces text classification experiments from paper descriptions, without access to reference code or repositories, through a pipeline that runs autonomously after a single human review checkpoint on the extracted configuration. Unlike prior AI Scientist systems, REPA targets the reproduction problem directly by deconstructing it as a four-stage process incorporating protocol extraction, input preparation, experiment generation, and evaluation. On a favorable set of ten well-documented text classification papers, REPA completes eight studies when instantiated with a GPT-4o backend, and five with Qwen3-Coder-30B. Compared to a replication rate of 0 via direct prompting, REPA establishes a performance ceiling for current LLM automation as of mid-2026 and underscores the importance of template scaffolding in scientific reproducibility.

## 1. Introduction

The volume of machine learning (ML) papers published annually exceeds the capacity of researchers available to independently verify them (Gundersen et al., 2025), with only 50% of highly-cited AI papers reproducible to any extent. Raff (2019) independently attempted to reimplement 255 ML papers without consulting author code, achieving a 63.5% success rate and finding that documentation quality predicts reproducibility. This gap motivates asking whether LLMs can automate the reproduction workflow.

Recent AI Scientist systems (Lu et al., 2024) demonstrate that LLMs can already operate as autonomous research agents: given a hypothesis, they generate code, execute experiments, interpret results, and write up findings end-to-end. However, existing systems either target *novel discovery* or evaluate from *provided repositories*. We target the valuable gap of addressing the verification problem of reproducing existing published results from paper descriptions.

REPA is structured as a general, four-stage framework applicable to empirical ML settings: protocol extraction parses a paper into a machine-readable specification of datasets, models, hyperparameters, and success criteria. Input preparation generates and executes a data download and preprocessing script. Experiment generation and execution produces a unified training script via LLM with an iterative self-correction loop of up to five repair attempts driven by error tracebacks, and reproduction evaluation applies a five-level statistical taxonomy followed by an LLM contextual layer that distinguishes genuine failures from constraint explained gaps. The extracted configuration is reviewed by a human before the remaining stages proceed; this single checkpoint is the pipeline's only manual step.

To demonstrate the feasibility of the REPA framework, we instantiate it in this demonstration paper in the domain of text classification. REPA generates PyTorch experiment code from a ∼5,000-token prompt of complete class definitions, supporting six classification model types including CNN, LSTM, DPCNN, and DAN.

Evaluated across ten text classification papers (2014–2022), REPA completes experiment generation for 8/10 papers with GPT-4o and 5/10 with Qwen3-Coder-30B.[1] When both backends succeed, reproduced accuracies differ by less than 1% on average, suggesting that template scaffolding rather than LLM capability determines reproduction outcome. Our contributions are: (1) the REPA framework, with a domain generic four stage architecture, (2) a five level statistical taxonomy separating methodological from infrastructural failures, and (3) a structured failure analysis identifying four failure classes previously uncharacterised in automated reproduction. We release REPA as an open benchmark for evaluating LLM-based reproducibility agents, providing a concrete framework and failure taxonomy to build on.[2]

[1]School of Computing, National University of Singapore (NUS), Singapore. Correspondence to: Nura Tamton <nurariaz@gmail.com>.

*Proceedings of the $43^{rd}$ International Conference on Machine Learning*, Seoul, South Korea. PMLR 306, 2026. Copyright 2026 by the author(s).

[1]Experiments executed January–March 2026. Results reflect GPT-4o and Qwen3-Coder-30B capability as of that date and may not generalise to later model versions.

[2]Code available at: https://github.com/nuratamton/ai-scientist-reproducibility

## 2. Related Work

**Reproducibility in ML:** Beyond the empirical reproduction rates noted above, Henderson et al. (2018) and Bouthillier et al. (2019) showed that intrinsic variance from hardware, library versions, and random seeds introduces non-trivial noise, even when code and data are shared, motivating multi-seed evaluation and tolerance-based taxonomy that we adopt.

**LLM code generation and self-repair:** LLMs have shown strong performance on standard programming benchmarks (Chen et al., 2021). Olausson et al. (2024) found that self-repair (correcting generated code from error traces) offers modest but inconsistent gains when repair cost is accounted for. They concluded that current models are held back by their inability to produce useful feedback on its own incorrect code.

**Autonomous scientific agents:** The AI Scientist (Lu et al., 2024) generates, executes, and writes up novel ML experiments end to end, targeting new discovery rather than verification of existing results. Agent Laboratory (Schmidgall et al., 2025) conducts literature review, experimentation, and report writing from a human seeded idea with optional human feedback at each stage. AIDE (Jiang et al., 2025) treats ML engineering as tree search over candidate code solutions and serves as the scaffold for several research agent benchmarks. SciAgents (Ghafarollahi & Buehler, 2025) chains role specialised LLMs over ontological knowledge graphs to generate hypotheses in materials science. MLAgentBench (Huang et al., 2023) benchmarks agents on 13 ML tasks with starter code provided, finding that GPT-4-level models succeed on 37.5% of tasks. Paper2Code (Seo et al., 2025) generates code repositories from paper descriptions using explicit multi-agent planning stages. More recently, PaperRepro (Zhang et al., 2026) uses a two-stage multi-agent pipeline for reproducibility assessment, but operates on the authors' released code and data rather than reconstructing experiments from the paper description alone.

While these systems form evidence of the burgeoning interest in such systems, none of the above systems target the specific problem of verifying a particular paper's reported numbers without access to reference code.

## 3. REPA Framework

### 3.1. General Framework

An automated reproduction attempt can be broken down into four problems in sequence: 1) extract what the paper claims to have done, 2) prepare the inputs the procedure requires, 3) execute the procedure, 4) and compare outputs against reported results. These four stages are domain-invariant and can be applied to any field, but what changes between domains is the implementation within each stage. Figure 5 shows the general architecture.

**Protocol extraction (PE)** parses a research paper into a machine readable specification: what inputs are required, what procedure is to be followed, what output constitutes a result, and what values count as successful reproduction. Output must be fully self-contained so that downstream stages can proceed without reference to the original paper. Figure 1 shows a representative PE output for Galke & Scherp (2022).

```
{
  "datasets": {
    "R8": {
      "source": "dxgp/R8",
      "num_classes": 8,
      "preprocessing": {
        "tokenization": "word_split",
        "lowercase": true,
        "vocab": "train_only",
        "min_freq": 2
      } }
    // ...R52, 20NG, MR (same schema)
  },
  "models": {
    "WideMLP_bow": {
      "type": "mlp",
      "architecture": {
        "hidden_size": 1024,
        "dropout": 0.5
      },
      "training": {"epochs": 100,
                   "lr": 0.001,
                   "optimizer": "adam",
                   "patience": 10}
    } },
  "evaluation": {"comparison_metric":
      "accuracy",
                 "scale": "percentage",
                     "num_seeds": 5},
  "baselines": {
    "R8": {"WideMLP_bow": {"accuracy":
        97.27, "std": 0.12}}
  } }
```

*Figure 1.* Abbreviated PE output for Galke & Scherp (2022). Two LLM calls at `temperature=0` extract datasets, models, hyperparameters, and baselines. A curated lookup table maps paper dataset names (e.g. `"MR"`) to HuggingFace identifiers (`rotten_tomatoes`), encoding domain knowledge the LLM cannot reliably infer. The config is human reviewed before execution proceeds to IP.

**Input Preparation (IP)** acquires and transforms whatever the procedure requires as inputs in a format the execution stage can consume without further interpretation. The implementation for this stage is domain specific: in ML, it involves dataset download and tokenisation; in computational biology, it involves sequence alignment and quality filtering; whereas, in numerical simulation, it involves parsing initial

conditions from the methods section. For example, in (Galke & Scherp, 2022), IP generates a `prepare_data.py` script that downloads `dxgp/R8` from HuggingFace, tokenises text, builds a vocabulary from training data only, and serialises train/val/test splits as `.pkl` files.

```
"""
Generate a Python data preparation script
for the {dataset_name} dataset.

PAPER INFO:
- Title: {paper_title}
- Dataset: {dataset_name}
- Source: {source}
- Classes: {num_classes}

COLUMN INFORMATION:
- Text column: {text_column}
- Label column: {label_column}
- Combine fields: {combine_fields}
- Subset: {subset}

REQUIREMENTS:
//... more details
Create train/val/test splits with
ROBUST error handling:
//... python sample
"""
```

*Figure 2.* Abbreviated IP prompt to the LLM

**Experiment Generation and Execution (EGE)** runs the procedure and collects outputs via a domain-independent contract: receive prepared inputs and the protocol specification, produce numeric outputs, and log failures. A *self-correction loop* captures errors and feeds tracebacks back to the LLM for repair, operating on error traces rather than semantic understanding of the code. For Kim (2014), GPT-4o generated a valid `experiment.py` on the first attempt (0 iterations), while Lin et al. (2017) required one repair after a shape mismatch in the attention layer.

**Reproduction Evaluation (RE)** compares outputs against claimed values and produces a verdict. The structure is invariant: compute a distance between reproduced and claimed results, assess whether it falls within an acceptable range, and account for known differences between the original setup and the reproduced one. What varies across domains is the distance metric, the acceptable range, and which setup differences must be accounted for. A two layer evaluation: statistical comparison followed by contextual judgment of constraint explained gaps is applicable across domains. For Galke & Scherp (2022) on R8, our reproduction achieves 97.1% against the paper's 97.27% ($\Delta = 0.17\%$), classified as *practical equivalence* under the five-level taxonomy (Section 4).

```
prompt = f"""
The following Python script has an
    error. Fix it carefully.

ERROR MESSAGE:
{error_msg}

{specific_guidance}  # error-specific
    hints, e.g. "saving train not test
    accuracy"

CURRENT CODE:
```python
{current_code}
```

OUTPUT REQUIREMENTS:
1. Fix the SPECIFIC error mentioned above
2. Maintain ALL existing functionality
3. Keep the same data loading format
4. Ensure you're saving TEST accuracy
    (not train accuracy)
5. Handle variable scoping correctly
    (load or pass required variables)
6. Output ONLY the corrected Python code
    with NO explanations

Generate the complete fixed code now:
"""
```

*Figure 3.* Self-correction prompt


```
{
  "dataset": "R8",
  "feature_type": "WideMLP_tfidf",
  "num_seeds": 5,
  "test_acc": {
    "mean": 97.2590223846505,
    "std": 0.1501295269543629,
    "values": [
      96.98492462311557,
      97.44175422567382,
      97.25902238465052,
      97.30470534490635,
      97.30470534490635
    ]
  },
  "metrics": {
    "accuracy": {
      "mean": 97.2590223846505,
      "std": 0.1501295269543629
    }
  }
}
```


*Figure 4.* Abbreviated EGE/RE output for Galke & Scherp (2022) (WideMLP-tfidf on R8): per-seed test accuracies aggregated into the `metrics.accuracy` structure the evaluation stage consumes. Mean 97.26% ($\pm$0.15) over five seeds.

### 3.2. Text Classification Instantiation

We instantiate this framework for text classification. A paper's arXiv abstract is parsed into a structured JSON configuration via two LLM calls at `temperature=0`: the first call extracting datasets, architectures, hyperparameters, and baselines and the second one extracting paper specific reproducibility criteria (primary metric and tolerance threshold). A curated lookup table of ∼20 entries maps paper dataset names to HuggingFace identifiers, encoding domain knowledge the LLM cannot reliably infer. Configurations are reviewed by a human before execution.

For experiment generation, the LLM generates a unified `experiment.py` script from a ∼5,000 token prompt consisting primarily of complete PyTorch class definitions. The script supports six model types: `mlp`, `fasttext`, `cnn` (Kim (2014)), `dpcnn`, `lstm`/`bilstm`, and `dan`. The self correction loop validates the script by running one epoch per model per dataset, feeding error tracebacks back to the LLM for up to five repair attempts. The LLM is stateless across calls so the full prior step script is injected as context at each iteration. Architectures outside the supported type system require extension of the prompt template. The validated script is then executed for each dataset–model–seed combination.

Evaluation uses the five-level statistical taxonomy described next in Section 4. Results that fail statistically are passed to an LLM contextual evaluation layer that judges whether the gap is explained by known constraints. A summary report and intervention log are generated at the end of each run.

## 4. Evaluation Framework

**Five-level statistical taxonomy.** Reproduction outcomes are classified as:

1. **Exact**: Difference $< 0.01\%$ representing numerically identical results.
2. **Statistical Equivalent**: Two One-Sided Tests (TOST) (Lakens, 2017) pass at $\alpha = 0.05$. Requires the paper to report both mean and standard deviation.
3. **Practical Equivalent**: Difference is within the paper-specific tolerance and the 95% confidence interval for the difference either contains zero or the effect size (Cohen's $d$) is small.
4. **Inconclusive**: The 95% CI contains zero but the difference exceeds the tolerance suggesting that there is not enough evidence to conclude either way.
5. **Failed**: Difference exceeds tolerance and the 95% CI excludes zero suggesting that the reproduction is statistically and practically different.

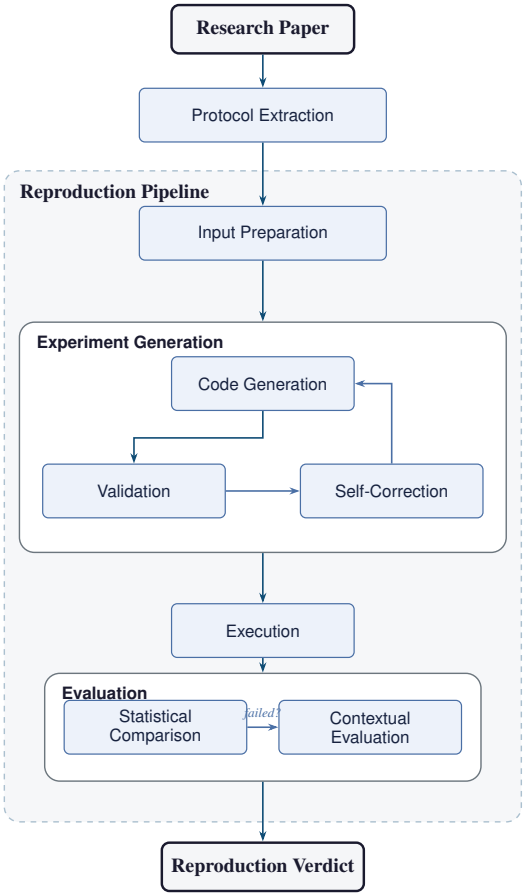


*Figure 5.* General four-stage framework for automated reproducibility. Solid boxes = concepts (inputs/outputs), shaded boxes = operators (processing stages), dashed borders = sub-components.

This distinguishes methodological failures (the model does not reproduce the paper's approach) from infrastructural limitations (the paper's approach cannot be reproduced within the pipeline's compute budget or model type system).

**LLM contextual evaluation:** When statistical evaluation classifies a result as failed, a second LLM call is made with a constraint report listing known limitations (embedding substitutions, dataset subsampling ratios, architecture approximations) and asks the LLM to judge whether these constraints fully explain the observed gap. This two layer evaluation avoids penalizing the pipeline for gaps that no automated system could close without additional resources.

## 5. Results

Table 1 summarises pipeline completion for each paper.

GPT-4o's failures were infrastructure events but Qwen's were all caused by generated code violating the required output format. In three of the five cases (DAN, Self-Attentive, HAN), Qwen saved results as `{best_val_acc: ...}` instead of the required `{metrics: {accuracy:`

| Paper | GPT-4o | | Qwen | |
|---|---|---|---|---|
| | Status | Iter. | Status | Iter. |
| Galke & Scherp (2022) | ✓ | 0 | ✓ | 1 |
| Kim (2014) | ✓ | 0 | ✓ | 0 |
| Joulin et al. (2017) | API err. | — | $5^\dagger$ | 5 |
| Johnson & Zhang (2017) | ✓ | 0 | ✓ | 0 |
| Zhang et al. (2015) | ✓ | 0 | $5^\dagger$ | 5 |
| Iyyer et al. (2015) | ✓ | 0 | $5^\dagger$ | 5 |
| Liu et al. (2016) | ✓ | 0 | ✓ | 0 |
| Lin et al. (2017) | ✓ | 1 | $5^\dagger$ | 5 |
| Yang et al. (2016) | ✓ | 1 | $5^\dagger$ | 5 |
| Lai et al. (2015) | Crash | 1 | ✓ | 0 |
| **Total** | **8/10** | | **5/10** | |

*Table 1.* Pipeline completion by paper and LLM backend. Iterations = self correction iterations before passing validation (0 = first-attempt success, $5^\dagger$ = all attempts failed).

`...}}` structure, despite the prompt containing an exact code block specifying the correct format. Self correction did not resolve any of Qwen's five failures: the loop produced superficially different code each iteration but never adopted the required `metrics.accuracy` structure which was a convergence failure in which the LLM applies syntactic variations without addressing the structural violation.

### 5.1. Reproduction Accuracy

**Galke & Scherp (2022):** Both backends reproduce R8 dataset within 1% (practical equivalence). GPT-4o reproduces R52 and MR dataset but Qwen's data preparation scripts failed on these datasets due to incorrect column access patterns, reducing its evaluation to three of five datasets. The 20 Newsgroups and Ohsumed datasets show 17–21% and 4–5% gaps respectively, consistent across both backends, indicating a systematic preprocessing issue rather than an LLM specific failure.

**Kim (2014):** The most extensively evaluated paper, with four model variants across six datasets. Pretrained embedding models on Subj achieve practical equivalence for both backends (<1% gap). The largest gaps occur on TREC (7–8% below the paper for pretrained models) and CNN-rand variants across all datasets (5–14% below). GPT-4o achieves practical equivalence on 6/24 experiments and Qwen on 7/24. With LLM contextual evaluation, both classified 10/24 experiments as reproduced. An anomaly in Qwen's results: all four CNN variants on CR (Customer Reviews dataset) produce 10–16% *above* the paper's reported values (e.g., 96.26% vs. 79.80% for CNN-rand) with high variance ($\pm$1.7–4.3%) due to data leakage. GPT-4o's CR results are 2–4% below the paper, consistent with evaluation protocol mismatch.

**Johnson & Zhang (2017):** All five datasets show substantially higher error rates than the paper across both backends (2.4–39.3% gaps). Both backends produce similar numbers on smaller datasets (AG News, DBpedia) and diverge on larger ones (Yahoo!: 49.3% vs. 63.2% error). The pipeline subsamples datasets to 100,000 examples so the large Yahoo! gap is attributable to aggressive subsampling of 1.4M training samples.

**Shared-backend comparison:** Four papers completed under both backends: (Galke & Scherp, 2022), (Kim, 2014), (Johnson & Zhang, 2017), and (Liu et al., 2016). On Kim 2014, the mean absolute difference between backends across 24 shared experiments is 0.67%. On DPCNN (Johnson & Zhang, 2017), differences are under 1.2% for four of five datasets. The Liu et al. (2016) results are nearly identical across backends (SST-2: 50.92% with zero variance; SUBJ: 48.35% with zero variance), indicating a shared training failure that both backends exhibit independently.

### 5.2. Failure Pattern Analysis

**Infrastructure-driven gaps:** Datasets with >100,000 training samples show consistently worse reproduction independent of LLM backend or architecture. Yahoo! Answers (1.4M→100k), Yelp Full (560k→100k), and 20 Newsgroups all produce results substantially below the paper. Aggressive subsampling likely degrades performance because models trained on a fraction of the data do not see sufficient examples of rare but discriminative patterns so the gap reflects data starvation rather than code generation failure. These gaps would likely close with longer timeouts or larger compute allocations.

**Architecture approximation failures:** Papers whose models were approximated rather than faithfully implemented show the largest accuracy gaps. HAN (hierarchical attention approximated as flat BiLSTM) produces random baseline results. TextRCNN (recurrent convolutional context approximated as plain BiLSTM) loses 3–7%. These failures reflect a known limitation of the pipeline's model type system, which requires explicit template support for each architecture.

**Non-learning models:** The Liu et al. (2016) results are pathological: SST-2 accuracy is 50.92% with zero variance across all seeds *and* both backends, SUBJ is 48.35% with zero variance. These are consistent with majority-class prediction. The consistency across both LLMs suggests the bug lies in the LSTM training loop that both models independently generate which is a shared failure mode of template guided code generation for this architecture class.

**Data leakage:** Qwen's anomalous CR results (96% vs. the paper's 80–85%, high variance) are diagnostic of evaluating on training data. This failure is particularly dangerous because the pipeline's validation checks do not catch it:

accuracy is high but not implausibly so (<99.5%). A shuffle test or held-out verification step would be required for automatic detection.

## 6. Discussion

**Reproduction rates:** Our pipeline achieves 80% completion (GPT-4o) on text classification papers. This is higher than PaperBench's best agent score of 21% (Starace et al., 2025), but the tasks are not directly comparable: PaperBench targets recent ICML 2024 papers with novel architectures, while our benchmark uses older, well-documented papers with standard architectures. Nevertheless, the comparison illustrates how much domain scoping and prompt scaffolding contribute to completion rates showing that a fully open-ended agent degrades when it cannot rely on prior knowledge of the target architecture.

**Generalisation:** The 80% completion rate is measured on a deliberately favorable sample of pre-2023 papers with standard architectures and clear documentation, and is not intended to generalise across eras or domains. We expect it to fall substantially on more recent or noisier work for two reasons. First, novel architectures fall outside the fixed model type system and require template extension — the mechanism behind the HAN and TextRCNN approximation failures. Second, sparsely documented papers stress protocol extraction, the stage most dependent on description quality. Extending REPA to post-2022 papers would therefore require dynamic architecture synthesis in place of a fixed type system, plus leakage detection in the evaluation layer.

**Documentation quality:** GPT-4o's two failures were infrastructure events unrelated to documentation, consistent with Raff (2019)'s finding that documentation predicts reproducibility. While generating code from descriptions closes the gap that code sharing is meant to address (Gundersen et al., 2025), our failure modes (architecture approximations, data leakage) show that automated generation introduces its own class of silent errors that human coders would avoid.

**Self correction:** Olausson et al. (2024) found that self repair gains are bottlenecked by the model's ability to provide accurate feedback on its own code. Our results provide a concrete instantiation of this in a scientific reproduction context: Qwen's five failures all exhibit a fixed point convergence pattern where the loop produces superficially different scripts each iteration but never diagnoses the structural output format violation. GPT-4o's higher first-attempt compliance rate means it rarely triggers the loop, suggesting that for structured output tasks, prompt engineering at generation time is more effective than relying on self correction.

**Prompt scaffolding:** Paper2Code (Seo et al., 2025) achieves strong code generation quality through explicit multi agent planning (roadmap, architecture diagram, then code). Our pipeline achieves comparable functional correctness on shared tasks through a different mechanism: a single, heavily-engineered prompt containing complete PyTorch class definitions. The backend similarity finding when both GPT-4o and Qwen generate valid code, reproduced accuracies differ by <1% on average. This suggests that once the structural constraints are satisfied, the code is functionally determined by the template, regardless of which LLM wrote it.

## 7. Conclusion

REPA demonstrates that LLM-assisted reproduction of ML experiments is feasible for well documented papers with standard architectures, achieving 80% completion with GPT-4o on text classification. The more significant finding is diagnostic: failure modes cluster into four reproducible categories – infrastructure constraints, architecture approximation, silent data leakage, and shared training bugs. This suggests that closing the reproducibility gap requires not just better LLMs, but better specified papers: precise architecture descriptions, dataset split protocols, and hardware budgets that automated systems can act on.

The backend equivalence result (<1% difference between GPT-4o and Qwen when both succeed) points toward a practical conclusion: **prompt scaffolding, not model capability, is currently the binding constraint**. Future work should focus on replacing the fixed model type system with dynamic architecture synthesis so that novel architectures need not be pre-templated, adding leakage detection (e.g. a shuffle or held-out check) to the evaluation layer, and extending the four-stage framework to domains beyond text classification. REPA provides the taxonomy and failure analysis needed to make that progress systematic.

**Limitations:** The pipeline requires human configuration review, is scoped to text classification with a fixed model type system, and results reflect GPT-4o and Qwen3-Coder-30B capability as of early 2026.

## Impact Statement

This work aims to support scientific reproducibility in machine learning. Automated reproduction pipelines like the one in this paper could reduce the human expert time and effort required to verify published research. Potential risk could include over reliance on automated verdict without any human reviews or silent failures such as data leakage that the pipeline does not currently detect. These limitations have been documented and it is recommended that the pipeline outputs are treated as a starting point for human verification rather than a definitive judgement.

## References


Bouthillier, X., Laurent, C., and Vincent, P. Unreproducible research is reproducible. In Chaudhuri, K. and Salakhutdinov, R. (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 725–734. PMLR, 09–15 Jun 2019. URL https://proceedings.mlr.press/v97/bouthillier19a.html.

Chen, M., Tworek, J., Jun, H., Yuan, Q., de Oliveira Pinto, H. P., Kaplan, J., Edwards, H., Burda, Y., Joseph, N., Brockman, G., Ray, A., Puri, R., Krueger, G., Petrov, M., Khlaaf, H., Sastry, G., Mishkin, P., Chan, B., Gray, S., Ryder, N., Pavlov, M., Power, A., Kaiser, L., Bavarian, M., Winter, C., Tillet, P., Such, F. P., Cummings, D., Plappert, M., Chantzis, F., Barnes, E., Herbert-Voss, A., Guss, W. H., Nichol, A., Paino, A., Tezak, N., Tang, J., Babuschkin, I., Balaji, S., Jain, S., Saunders, W., Hesse, C., Carr, A. N., Leike, J., Achiam, J., Misra, V., Morikawa, E., Radford, A., Knight, M., Brundage, M., Murati, M., Mayer, K., Welinder, P., McGrew, B., Amodei, D., McCandlish, S., Sutskever, I., and Zaremba, W. Evaluating large language models trained on code, 2021. URL https://arxiv.org/abs/2107.03374.

Galke, L. and Scherp, A. Bag-of-words vs. graph vs. sequence in text classification: Questioning the necessity of text-graphs and the surprising strength of a wide MLP. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 4038–4051, 2022.

Ghafarollahi, A. and Buehler, M. J. SciAgents: Automating scientific discovery through bioinspired multi-agent intelligent graph reasoning. *Advanced Materials*, 37(22):2413523, 2025. doi: https://doi.org/10.1002/adma.202413523. URL https://advanced.onlinelibrary.wiley.com/doi/abs/10.1002/adma.202413523.

Gundersen, O. E., Cappelen, O., Mølnå, M., and Nilsen, N. G. The unreasonable effectiveness of open science in AI: A replication study. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pp. 26211–26219, 2025.

Henderson, P., Islam, R., Bachman, P., Pineau, J., Precup, D., and Meger, D. Deep reinforcement learning that matters. In *Proceedings of the AAAI conference on artificial intelligence*, 2018.

Huang, Q., Vora, J., Liang, P., and Leskovec, J. MLAgentBench: Evaluating language agents on machine learning experimentation. *arXiv preprint arXiv:2310.03302*, 2023.

Iyyer, M., Manjunatha, V., Boyd-Graber, J., and Daumé III, H. Deep unordered composition rivals syntactic methods for text classification. In *Proceedings of the 53rd annual meeting of the association for computational linguistics and the 7th international joint conference on natural language processing (volume 1: Long papers)*, pp. 1681–1691, 2015.

Jiang, Z., Schmidt, D., Srikanth, D., Xu, D., Kaplan, I., Jacenko, D., and Wu, Y. AIDE: AI-driven exploration in the space of code, 2025. URL https://arxiv.org/abs/2502.13138.

Johnson, R. and Zhang, T. Deep pyramid convolutional neural networks for text categorization. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 562–570, 2017.

Joulin, A., Grave, E., Bojanowski, P., and Mikolov, T. Bag of tricks for efficient text classification. In Lapata, M., Blunsom, P., and Koller, A. (eds.), *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pp. 427–431, Valencia, Spain, April 2017. Association for Computational Linguistics. URL https://aclanthology.org/E17-2068/.

Kim, Y. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pp. 1746–1751, 2014.

Lai, S., Xu, L., Liu, K., and Zhao, J. Recurrent convolutional neural networks for text classification. In *Proceedings of the AAAI conference on artificial intelligence*, 2015.

Lakens, D. Equivalence tests: A practical primer for $t$ tests, correlations, and meta-analyses. *Social Psychological and Personality Science*, 8(4):355–362, 2017.

Lin, Z., Feng, M., Santos, C. N. d., Yu, M., Xiang, B., Zhou, B., and Bengio, Y. A structured self-attentive sentence embedding. In *International Conference on Learning Representations (ICLR)*, 2017.

Liu, P., Qiu, X., and Huang, X. Recurrent neural network for text classification with multi-task learning. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 2873–2879, 2016.

Lu, C., Lu, C., Lange, R. T., Foerster, J., Clune, J., and Ha, D. The AI Scientist: Towards fully automated open-ended scientific discovery, 2024. URL https://arxiv.org/abs/2408.06292.

Olausson, T. X., Inala, J. P., Wang, C., Gao, J., and Solar-Lezama, A. Is self-repair a silver bullet for code generation?, 2024. URL https://arxiv.org/abs/2306.09896.

Raff, E. A step toward quantifying independently reproducible machine learning research. *Advances in Neural Information Processing Systems*, 32, 2019.

Schmidgall, S., Su, Y., Wang, Z., Sun, X., Wu, J., Yu, X., Liu, J., Moor, M., Liu, Z., and Barsoum, E. Agent Laboratory: Using LLM agents as research assistants. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pp. 5977–6043. Association for Computational Linguistics, 2025.

Seo, M., Baek, J., Lee, S., and Hwang, S. J. Paper2Code: Automating code generation from scientific papers in machine learning, 2025. URL https://arxiv.org/abs/2504.17192.

Starace, G., Jaffe, O., Sherburn, D., Aung, J., Chan, J. S., Maksin, L., Dias, R., Mays, E., Kinsella, B., Thompson, W., Heidecke, J., Glaese, A., and Patwardhan, T. PaperBench: Evaluating AI's ability to replicate AI research, 2025. URL https://arxiv.org/abs/2504.01848.

Yang, Z., Yang, D., Dyer, C., He, X., Smola, A., and Hovy, E. Hierarchical attention networks for document classification. In *Proceedings of the 2016 conference of the North American chapter of the association for computational linguistics: human language technologies*, pp. 1480–1489, 2016.

Zhang, L., Xia, T., Piao, J., Cui, L., and Li, Y. PaperRepro: Automated computational reproducibility assessment for social science papers. *arXiv preprint arXiv:2603.00058*, 2026.

Zhang, X., Zhao, J., and LeCun, Y. Character-level convolutional networks for text classification. *Advances in neural information processing systems*, 28, 2015.