



Lightning AI

What Would a PyTorch Expert Say?

Turning tribal AI knowledge into training data



From the creators of PyTorch Lightning. Turn ideas into AI, Lightning fast.



Open source

PyTorch Lightning,
LitData, LitGPT, LitServe,
Lightning Thunder,
TorchMetrics, and more



The AI cloud

Build, train, and deploy
your own models in
Lightning Studios + Jobs



Compute at scale

36,000+ NVIDIA H200,
B200, and GB300 GPUs

Own your AI, don't rent it



Our task



GOAL

Post-train an efficient AI expert model for customer workflows



APPROACH

Use heterogeneous data sources that human experts actually use and contribute to, encouraging the model to mimic the expert



bilal

@bilaltwovec



sorta crazy how they just distilled my goat ptrblck 🙄



Liv



@livgorton · 15h

sort of crazy how often i read the pytorch docs like 18 months ago i miss them

4:24 AM · Oct 20, 2026 · **468** Views



Code fluency $\neq$ AI expertise



Software engineering

Starts from:

A specified issue

Correct looks like:

Whatever the repo and its tests already define

Today:

Best model resolves ~65% of fresh issues (SWE-Rebench)

<https://swe-rebench.com/>



AI research engineering

Starts from:

A paper or an idea

Correct looks like:

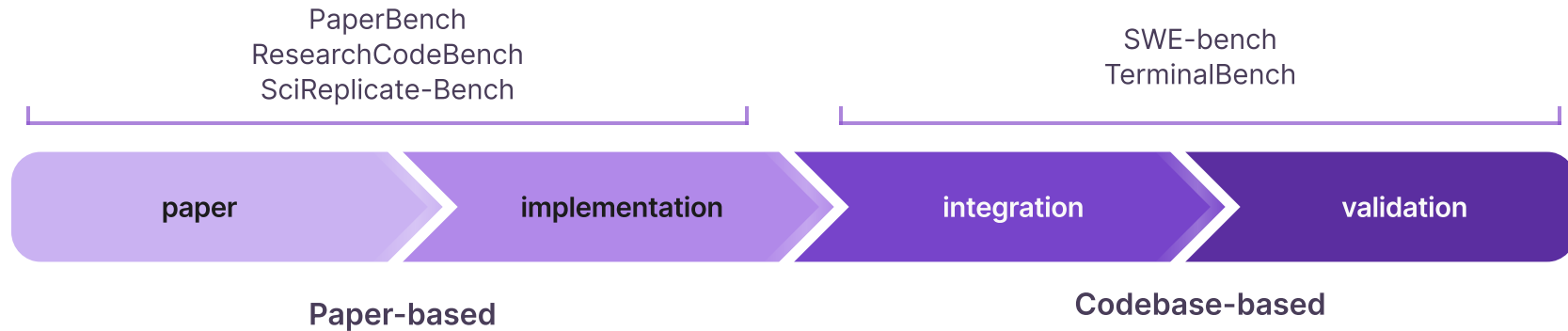
Faithful to the paper and working in the real system

Today:

Best model implements <40% of novel-paper code (ResearchCodeBench)

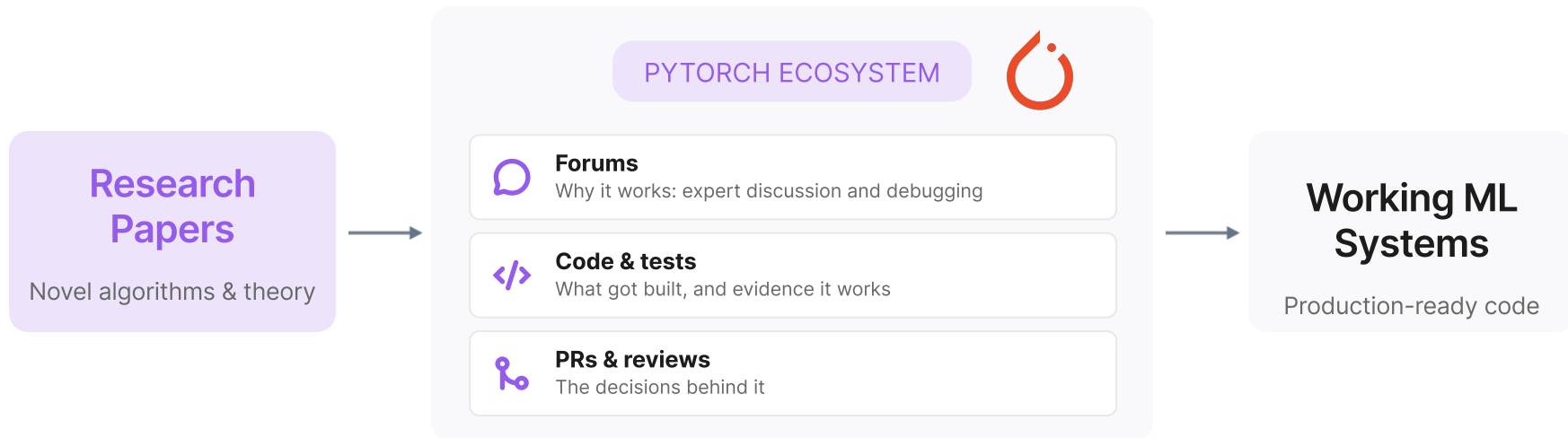
<https://researchcodebench.github.io/>

Existing benchmarks cover parts of the research workflow





Research becomes code in the PyTorch ecosystem



The data exists, but it is scattered across the ecosystem



PyTorch ecosystem

Modeling

Computer Vision



Specialized



AI for Science



Language



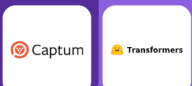
Medical & Biology



Multimodal



Adversarial



Quantum



RL



Training

General



TorchJD



TorchSurv



Privacy



Federated



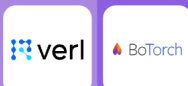
RL



Distributed



Probabilistic



Graph

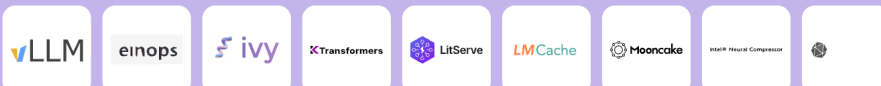


Multimodal



Optimizations

General



TensorDict



TokenSpeed



SGL



XGrammar



Compilers



MLOps



FEAST



Distributed





Data sources



Forums



Github PRs



arXiv papers



Forum data is rich but messy

user_42

Why does my training loop crash after the first epoch?

```
loss = 0
for epoch in range(10):
    loss += model(x).sum()
    loss.backward()
```

broken code

— user_42 · [user_42@example.com](#)

personal info

user_17

```
optimizer.zero_grad()
...
```

partial snippet

user_08

Check out my course for the full answer!

promotional



Scrub personal info



Remove promotional posts



Filter for quality (LLM judge)



~40K threads



Initial results

PyTorch Diagnostic Suite Task

Given: broken code

```
import torch
x = torch.tensor([2.], requires_grad=True)
loss = x.detach().square().sum()
loss.backward()
```



Expected: a complete fix that runs

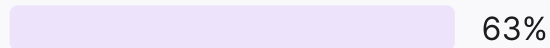
```
import torch
x = torch.tensor([2.], requires_grad=True)
loss = x.square().sum()
loss.backward()
```

Qwen 3.8-27B, % of fixes that run and pass

Base model



+ forum retrieval



SFT on forum data





Forum synthetic data generation



Human

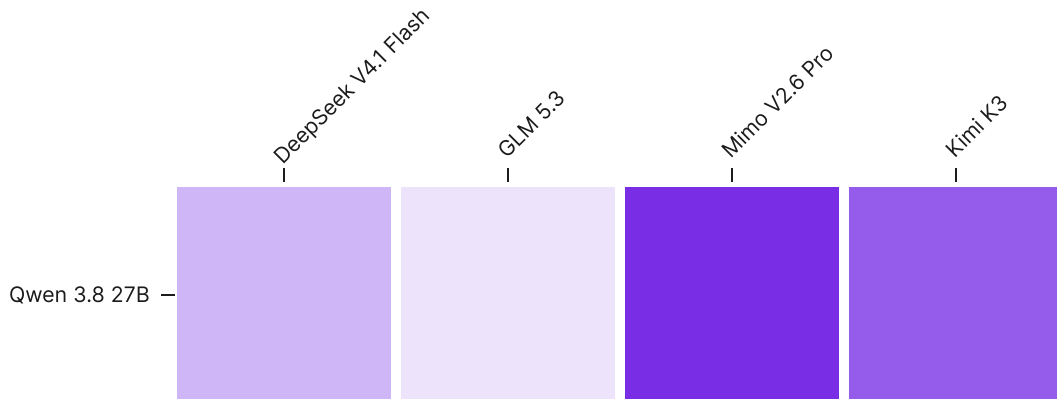
What does “normal random sampling” mean in LLM text generation?

#225384

In PyTorch-based LLM stacks, "sampling" does concretely mean `torch.multinomial(F.softmax(logits, -1), 1)` in the common cases — but that's a convention, not a spec. If your target is a well-defined object of study, define your own sampler explicitly, because multinomial's finite-precision behavior is an implementation detail, not a contract.



LLM





Data sources



Forums



Github PRs



arXiv papers



Mining GitHub PRs that implement arXiv papers

Example: T-LRU (NeurIPS 2025) → SGLang PR #34012



[Submitted on 16 Oct 2025]

Tail-Optimized Caching for LLM Inference

Wenxin Zhang, Yueying Li, Ciamac C. Moallemi, Tianyi Peng

```
python/sglang/srt/nn_cache/evict_policy.py  +35 00000
69 + def get_priority(self, node: TreeNode) -> Tuple[int, float]:
70 +     # node._tlru_history_len is the branch's high-water depth, i.e. the
71 +     # paper's L, and deliberately does not shrink when the tail is trimmed.
72 +     # Deriving L from what is still resident instead would leave the
73 +     # shortened conversation over budget on the next pass too, and T-LRU
74 +     # would walk it down to nothing rather than stopping after
75 +     # (threshold - Q_hat) tokens.
76 +     budget = max(
77 +         node._tlru_history_len + self.next_prompt_estimate - self.threshold, 0
78 +     )
79 +     cached_without_this_node = node._tlru_cached_prefix_len - len(node.key)
80 +     tel_safe = cached_without_this_node >= budget
81 +     return (-1 if tel_safe else 0, node.last_access_time)
82 +
```

```
test/registered/unit/nn_cache/test_tlru_eviction_policy.py  +208 00000
174 + def test_budget_clamps_at_zero():
175 +     """A budget below zero must mean nothing needs caching, not wrap around."""
176 +     s = strategy()
177 +     node = FakeNode(_tlru_cached_prefix_len=PAGE, key_len=PAGE, _tlru_history_len=0)
178 +     assert is_tel_safe(s, node)
```

sgl-project / sglang

Add Agentic-Aware Tail-Optimized LRU eviction to the unified radix cache #34012

Isabock merged 9 commits into sgl-project:main from y13469:dev-dsv4-gb308-tlru on Sep 17

y13469 commented on Aug 7 · edited by github-actions Bot · Contributor

Motivation

Tail-Optimized LRU (T-LRU, [NeurIPS 25](#)) keeps only as much of a conversation cached as its next prefix needs to stay under a TTFT SLO budget, and frees the rest of the tail before falling back to recency order, so newer requests can be admitted. On agentic traces the tail is what a plain LRU protects at the expense of every other conversation's tail latency by up to 43%.

----- The spec

----- The gold patch

----- The reward

----- Maintainer-vetted



Finding PRs that implement papers

1 Starting dataset of PR diffs from target repositories ~80,000

2 Link to an arXiv paper 950

3 Have been merged 834

4 Cite exactly only one paper 512

5 Attempt to implement the paper, not just cite it 435

6 Have test files 206



How we clean and vet each PR



01



02



03



04

Quality check

Is the implementation faithful?

Check against the paper

Group & patch

Create one gold patch

Combine related PRs + fill gaps

Extend tests

Make tests a real signal

Add coverage for the paper's method

Pin environment

Make it reproducible

Lock dependencies + validate



What a test run taught us

Implementing isn't integrating

keep $L + \hat{Q} - \xi$

 agent

 real PR

Older radix cache

Unified radix cache

Two different questions for grading

f_x

Is the math right?

Unit tests, LLM judge



Does it work in the system?

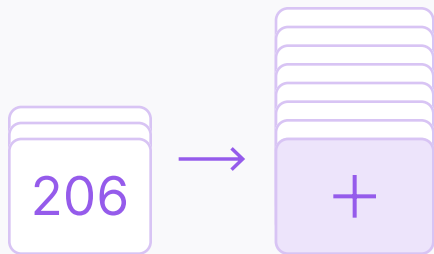
Integration tests, rubric items



Next steps

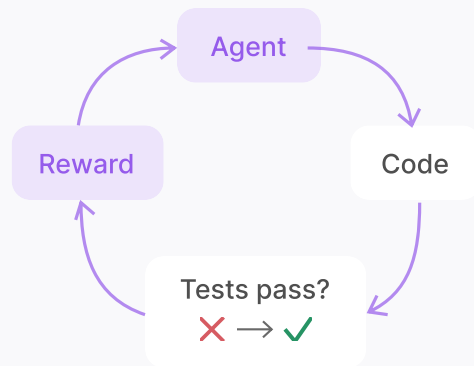
1

Scale the dataset



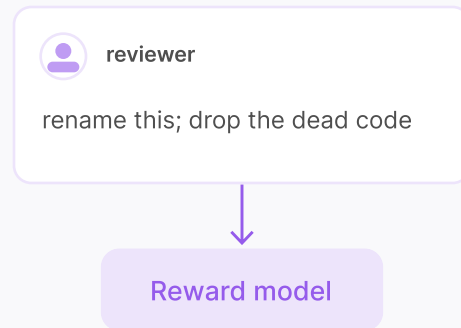
2

RL with tests as reward



3

Learn from reviewers





Our Takeaways

1

The expertise is already out there

In this community's PRs, reviews and tests

2

Finding it is easy. Trusting it is hard.

~80,000 PRs → 206 candidates, before cleaning

3

Research code needs more than one kind of check

Is the math right? Does it work in the system?

Thank you, PyTorch community



Our team



Daniela Dapena



Liana Koleva



Teja Pulagam



Michael Beidler



Yash Kankariya



Get in touch

liana@lightning.ai

teja@lightning.ai

Join us

lightning.ai/careers



What Would a PyTorch Expert Say? Turning Tribal AI Knowledge
into a Model