Observations and speculations on continual learning in LLMs

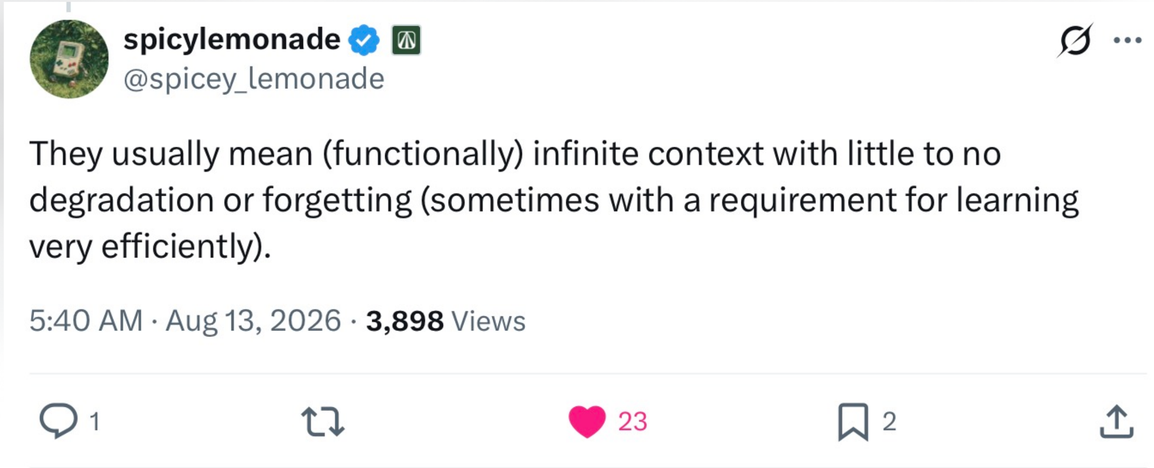
Hiranmay Darshane for the inaugural Mumbai alphaXiv paper club meet.

For detailed references, comments and further reading:
hiranmay.com/cl-talk

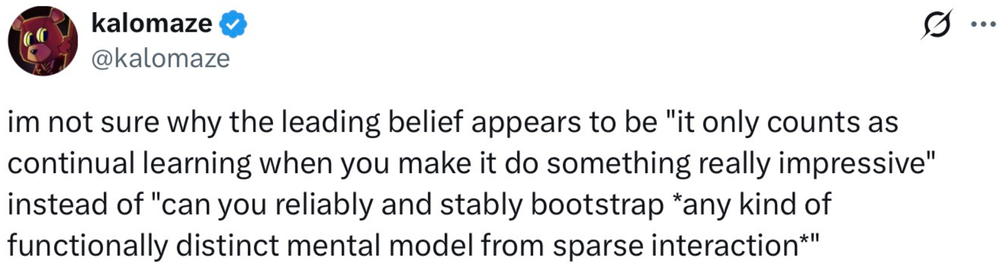
super hard to define in isolation! too many interpretations. instead let’s look at thoughts from those in the field and try to define it in terms of problems, shortcomings and desiderata:

- scaling law between time spent in an env vs perf gains





- scaling law between time spent in an env vs perf gains
- operating over ultra-long contexts
- minimal forgetting



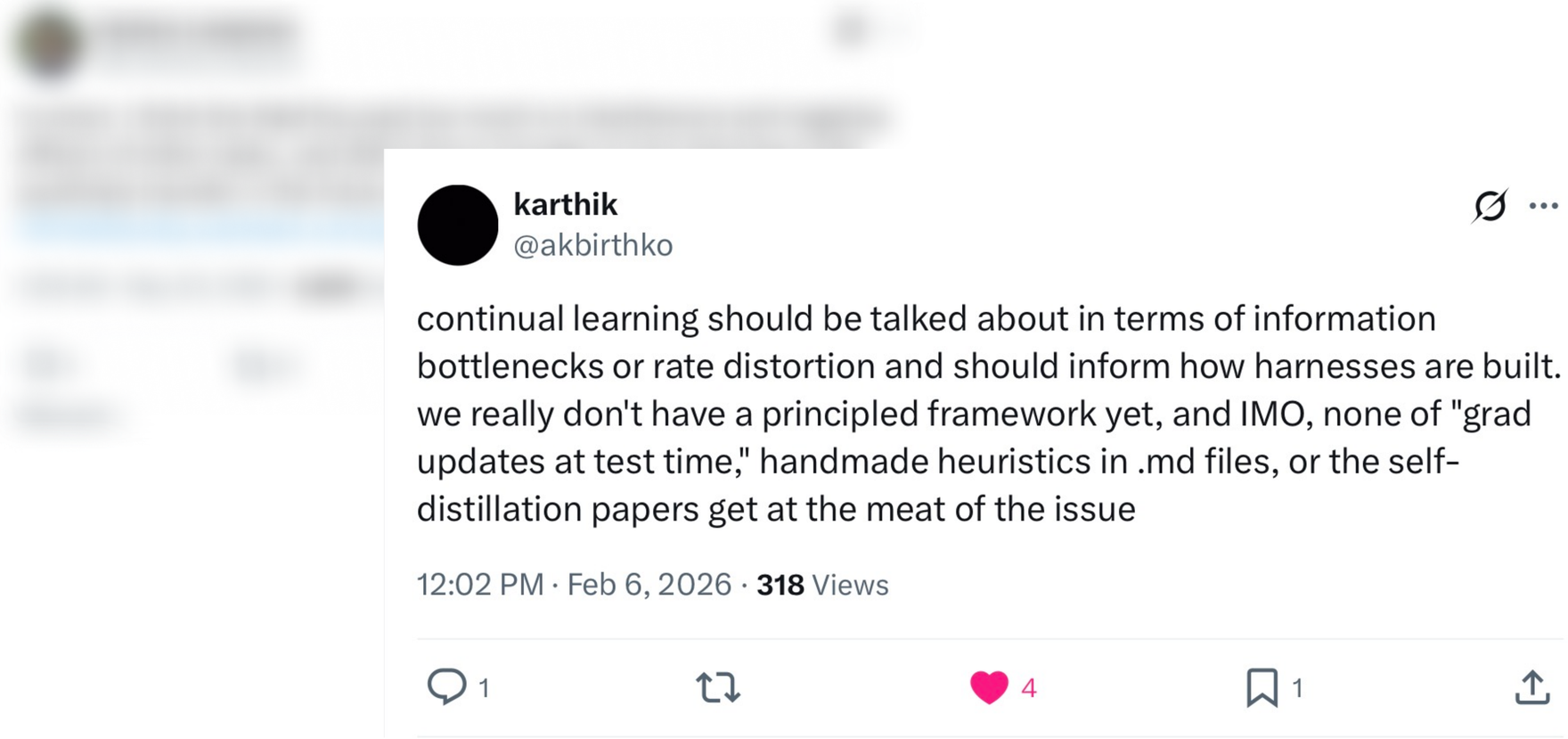
- scaling law between time spent in an env vs perf gains
- operating over ultra-long contexts
- minimal forgetting
- learning from “very little”
- learning to develop and think in terms of newer abstractions

the problem has very certainly been misconstrued to a great degree. it’s time to track back and reframe it s.t. it is tractable!



- scaling law between time spent in an env vs perf gains
- operating over ultra-long contexts
- minimal forgetting
- learning from “very little”
- learning to develop and think in terms of newer abstractions
- maximising transfer between learnt tasks

the problem has very certainly been misconstrued to a great degree. it’s time to track back and reframe it s.t. it is tractable!



karthik
@akbirthko

continual learning should be talked about in terms of information bottlenecks or rate distortion and should inform how harnesses are built. we really don't have a principled framework yet, and IMO, none of "grad updates at test time," handmade heuristics in .md files, or the self-distillation papers get at the meat of the issue

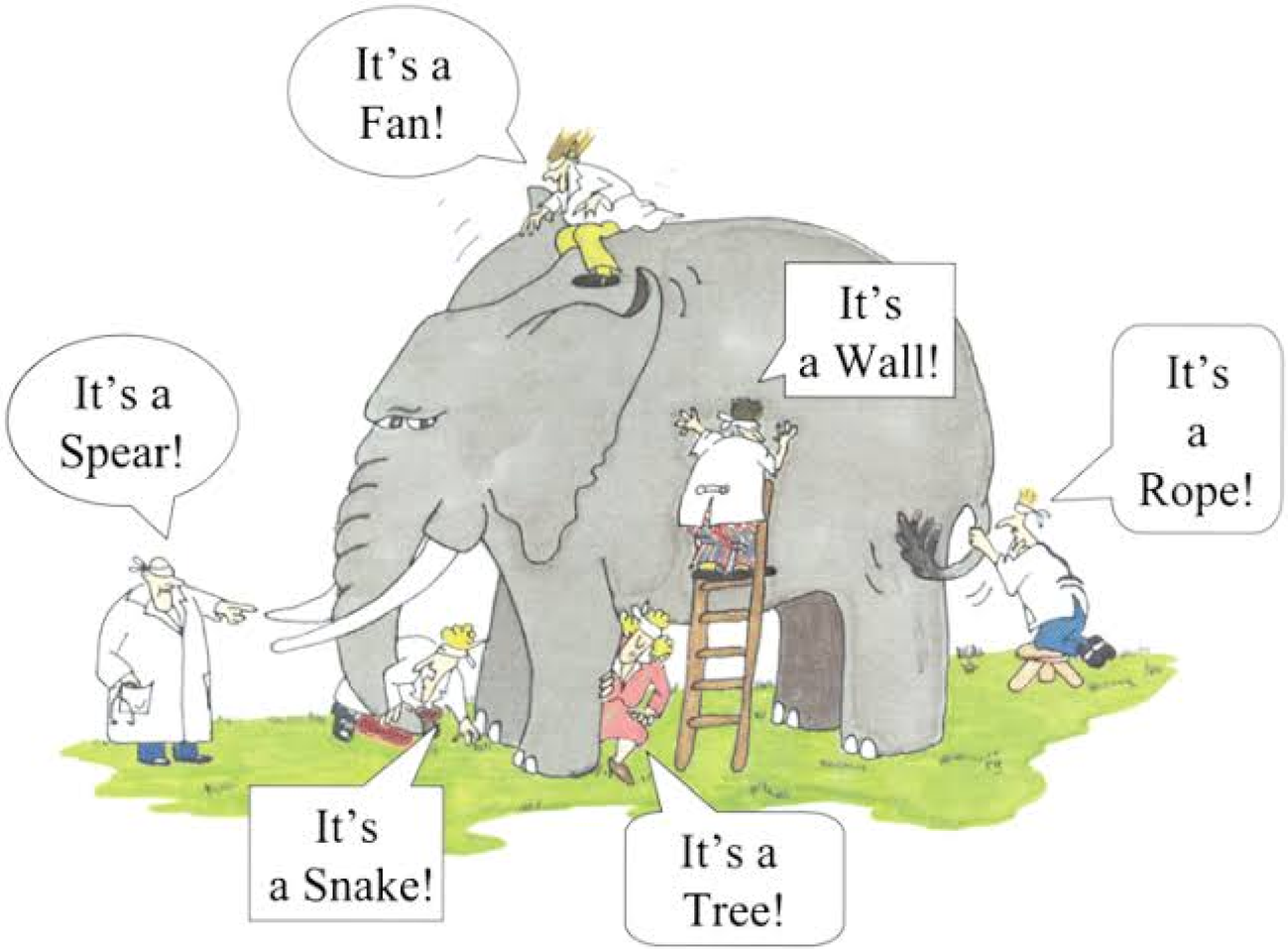
12:02 PM · Feb 6, 2026 · **318** Views

1 4 1

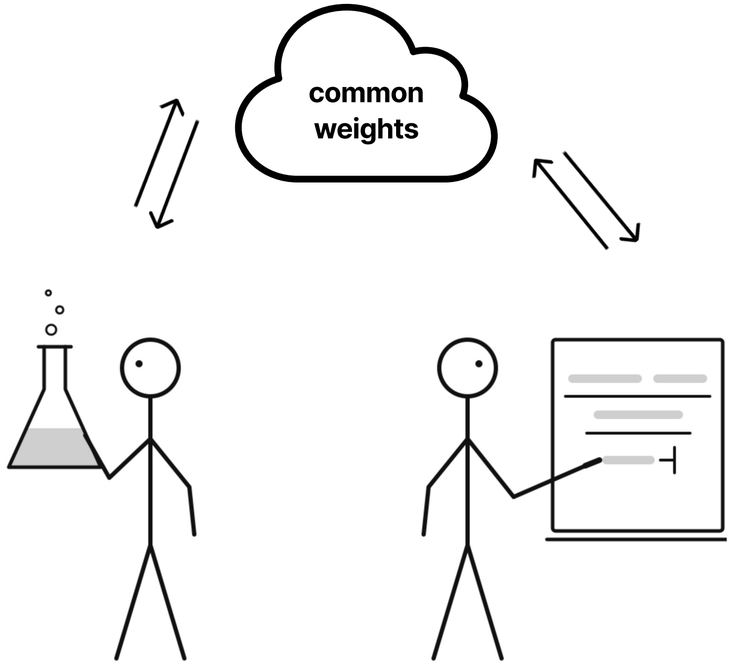
- scaling law between time spent in an env vs perf gains
- operating over ultra-long contexts
- minimal forgetting
- learning from “very little”
- learning to develop and think in terms of newer abstractions
- maximising transfer between learnt tasks
- deciding what to remember (in terms of future expectations)

There is no one-shot solution in isolation.
And that is amazing!

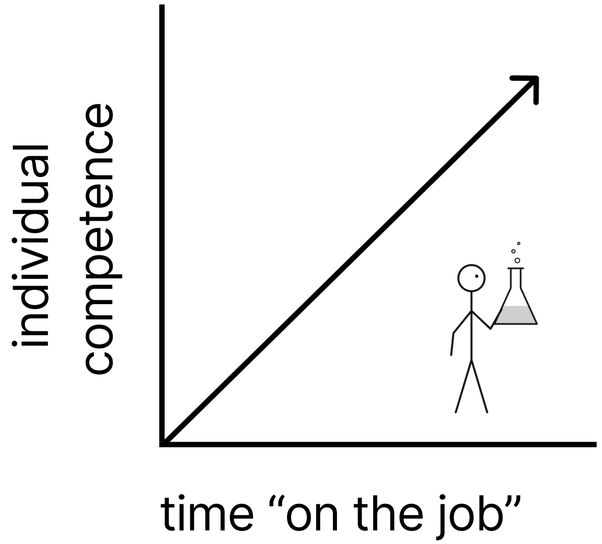
What we desire	Already an open problem called
Ultra-long contexts	Long-context modelling (either via architecture or context management)
Minimal forgetting	Plasticity, capacity
Learning from very little	Sample efficiency
New abstractions	Generalisation
Transfer between tasks	Transfer learning
Deciding what to remember	Memory and compression, long-horizon tasks


It's a Fan!
It's a Spear!
It's a Wall!
It's a Rope!
It's a Snake!
It's a Tree!

a very common misframing:



vs



An economist’s view of continual learning

Pretraining is a central planner. It decides what matters by how often the world wrote it down (or how much data teams at labs or their vendors are appreciative of it).

Real-world deployment lives in the long tail: sparse, local, often proprietary knowledge that never reaches pretraining.
Hayek’s “knowledge of the particular circumstances of time and place” that a central planner can’t foresee.

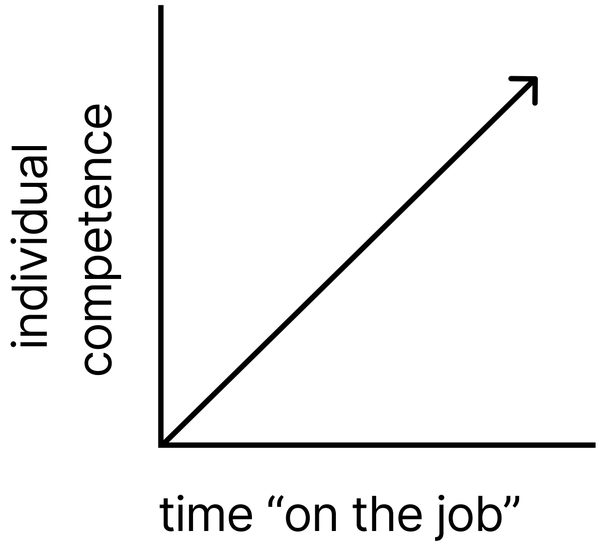
Continual learning is most needed there. Everything else, the next pretrain *can* absorb.

F. A. Hayek
“The Use of Knowledge in Society”

“Every AI agent today is an amnesiac intern.”
Roughly-stated goal: an intern that gets better the longer it works somewhere.

slightly better criteria:

1. Correctly choose what is to be learnt, and learn it at the right level of abstraction with as few samples as possible
2. Don’t overwrite what the base model knew
3. Don’t forget what the earlier updates added
4. Don’t degrade general capability, including the potential of learning the next thing.



criteria 2–4 after Charlie O’Neill
“Trying to actually define continual learning”

Pretraining as a successful “central planner”

Pretraining decides what to learn, and generalises from it
remarkably well, without anyone telling it what matters.

A deployed model never gets to make that decision.
Whoever writes its prompt, data or reward makes it instead.
The “learning” part isn’t exactly in the hands of the learner.

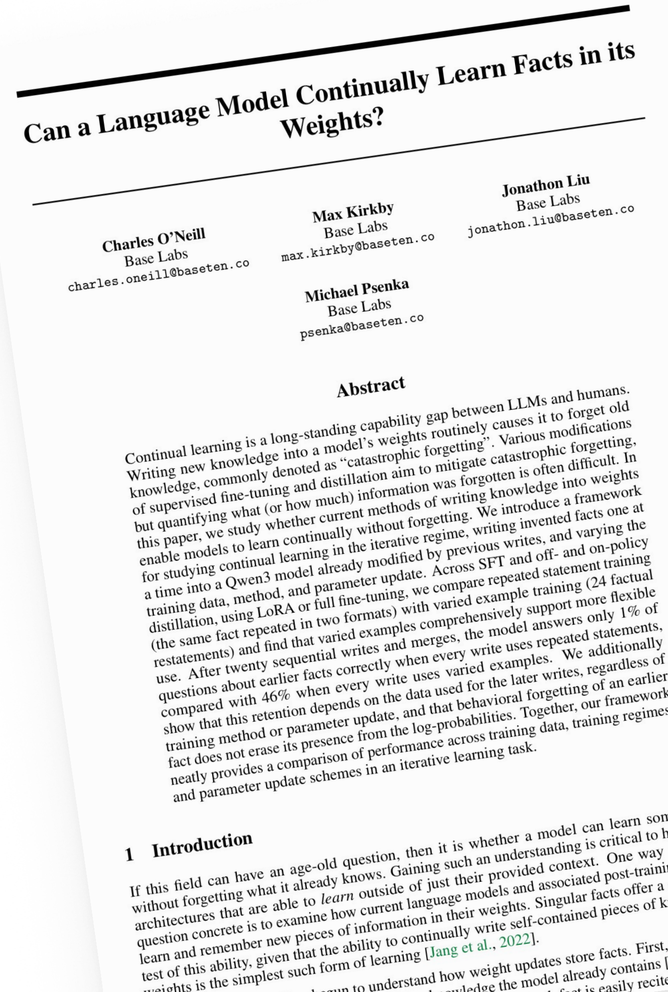
First: why does it work for pretraining?

Fun experiment: Invent facts no model has ever seen, and try to teach them.

“Zorvathine is a metal that melts when cooled below −10 °C.”

Finetune on it. Then ask:
“What is Zorvathine’s phase behavior?”
“Would a Zorvathine vessel stay solid while holding liquid nitrogen?”
“How does Zorvathine respond to cooling, unlike an ordinary metal?”

O’Neill et al.
“Can a Language Model Learn Facts Continually in Its Weights?”



Can a Language Model Continually Learn Facts in its Weights?

Charles O’Neill
Base Labs
charles.oneill@baseten.co

Max Kirkby
Base Labs
max.kirkby@baseten.co

Jonathon Liu
Base Labs
jonathon.liu@baseten.co

Michael Psenka
Base Labs
psenka@baseten.co

Abstract

Continual learning is a long-standing capability gap between LLMs and humans. Writing new knowledge into a model’s weights routinely causes it to forget old knowledge, commonly denoted as “catastrophic forgetting”. Various modifications of supervised fine-tuning and distillation aim to mitigate catastrophic forgetting, but quantifying what (or how much) information was forgotten is often difficult. In this paper, we study whether current methods of writing knowledge into weights enable models to learn continually without forgetting. We introduce a framework for studying continual learning in the iterative regime, writing invented facts one at a time into a Qwen3 model already modified by previous writes, and varying the training data, method, and parameter update. Across SFT and off- and on-policy distillation, using LoRA or full fine-tuning, we compare repeated statement training (the same fact repeated in two formats) with varied example training (24 factual restatements) and find that varied examples comprehensively support more flexible use. After twenty sequential writes and merges, the model answers only 1% of questions about earlier facts correctly when every write uses repeated statements, compared with 46% when every write uses varied examples. We additionally show that this retention depends on the data used for the later writes, regardless of training method or parameter update, and that behavioral forgetting of an earlier fact does not erase its presence from the log-probabilities. Together, our framework neatly provides a comparison of performance across training data, training regimes and parameter update schemes in an iterative learning task.

1 Introduction

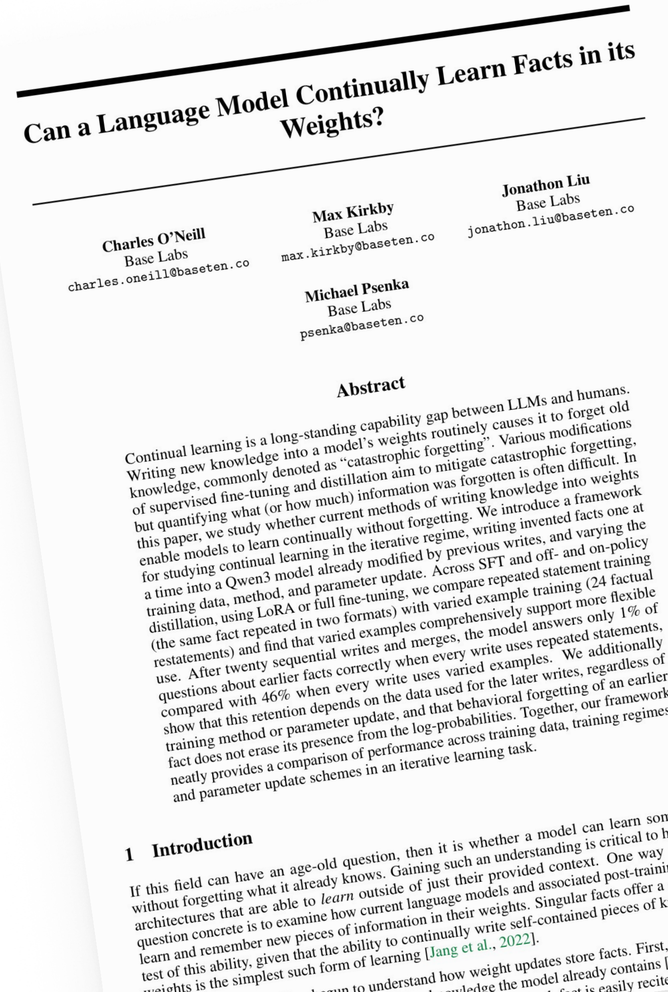
If this field can have an age-old question, then it is whether a model can learn som without forgetting what it already knows. Gaining such an understanding is critical to h architectures that are able to *learn* outside of just their provided context. One way question concrete is to examine how current language models and associated post-traini learn and remember new pieces of information in their weights. Singular facts offer a test of this ability, given that the ability to continually write self-contained pieces of k weights is the simplest such form of learning [Jang et al., 2022].

The model fails to correctly answer derived questions, though correctly recites the fact itself. Gap(fact-recitation, generalisation) ~= 26 points. (persists across model scales i.e. Qwen 4B and 8B)

But! Write 24 examples of the same fact, and the gap between recitation and generalising reduces to ~3 points.

What just happened?

O'Neill et al.
"Can a Language Model Learn Facts Continually in Its Weights?"



Can a Language Model Continually Learn Facts in its Weights?

Charles O'Neill
Base Labs
charles.oneill@baseten.co

Max Kirkby
Base Labs
max.kirkby@baseten.co

Jonathon Liu
Base Labs
jonathon.liu@baseten.co

Michael Psenka
Base Labs
psenka@baseten.co

Abstract

Continual learning is a long-standing capability gap between LLMs and humans. Writing new knowledge into a model's weights routinely causes it to forget old knowledge, commonly denoted as "catastrophic forgetting". Various modifications of supervised fine-tuning and distillation aim to mitigate catastrophic forgetting, but quantifying what (or how much) information was forgotten is often difficult. In this paper, we study whether current methods of writing knowledge into weights enable models to learn continually without forgetting. We introduce a framework for studying continual learning in the iterative regime, writing invented facts one at a time into a Qwen3 model already modified by previous writes, and varying the training data, method, and parameter update. Across SFT and off- and on-policy distillation, using LoRA or full fine-tuning, we compare repeated statement training (the same fact repeated in two formats) with varied example training (24 factual restatements) and find that varied examples comprehensively support more flexible use. After twenty sequential writes and merges, the model answers only 1% of questions about earlier facts correctly when every write uses repeated statements, compared with 46% when every write uses varied examples. We additionally show that this retention depends on the data used for the later writes, regardless of training method or parameter update, and that behavioral forgetting of an earlier fact does not erase its presence from the log-probabilities. Together, our framework neatly provides a comparison of performance across training data, training regimes and parameter update schemes in an iterative learning task.

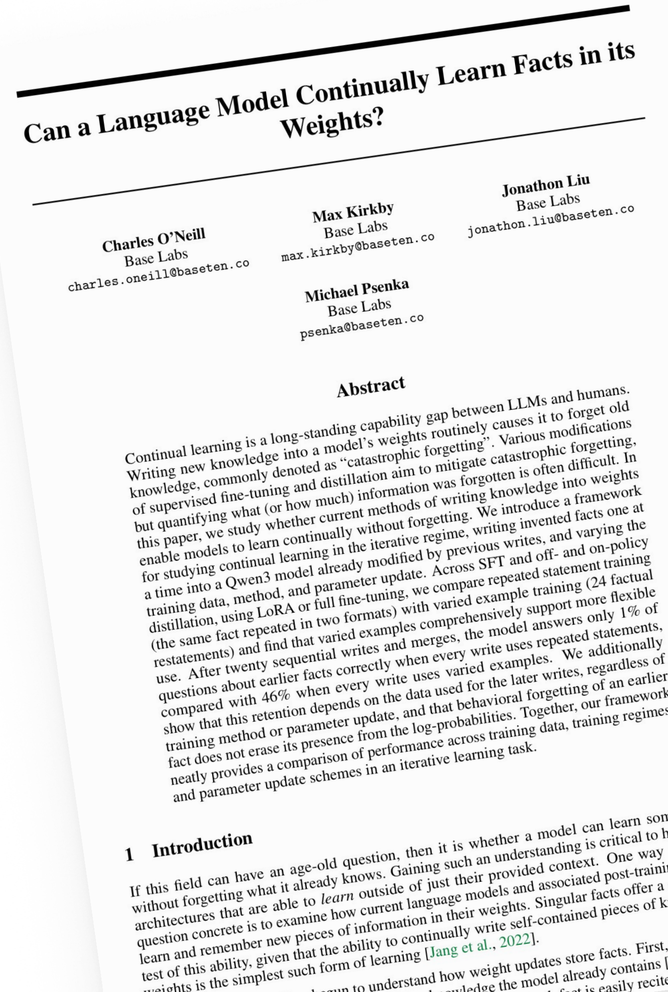
1 Introduction

If this field can have an age-old question, then it is whether a model can learn som without forgetting what it already knows. Gaining such an understanding is critical to h architectures that are able to *learn* outside of just their provided context. One way question concrete is to examine how current language models and associated post-traini learn and remember new pieces of information in their weights. Singular facts offer a test of this ability, given that the ability to continually write self-contained pieces of k weights is the simplest such form of learning [Jang et al., 2022].

Let’s add to the puzzlingness.

Which matters more for remembering prior facts?

You teach the model one fact, then ten more. You can give it varied examples of the **first fact**, or varied examples of the **next ten**. What would you choose?



Can a Language Model Continually Learn Facts in its Weights?

Charles O’Neill
Base Labs
charles.oneill@baseten.co

Max Kirkby
Base Labs
max.kirkby@baseten.co

Jonathon Liu
Base Labs
jonathon.liu@baseten.co

Michael Psenka
Base Labs
psenka@baseten.co

Abstract

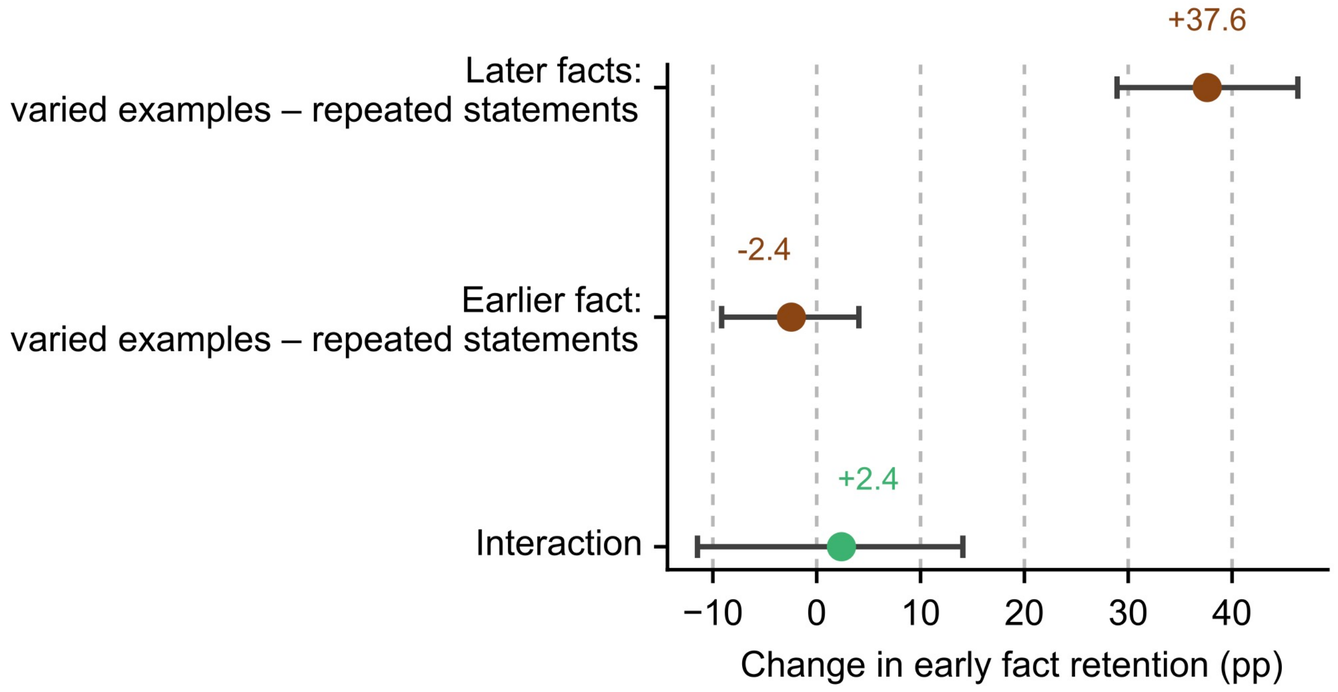
Continual learning is a long-standing capability gap between LLMs and humans. Writing new knowledge into a model’s weights routinely causes it to forget old knowledge, commonly denoted as “catastrophic forgetting”. Various modifications of supervised fine-tuning and distillation aim to mitigate catastrophic forgetting, but quantifying what (or how much) information was forgotten is often difficult. In this paper, we study whether current methods of writing knowledge into weights enable models to learn continually without forgetting. We introduce a framework for studying continual learning in the iterative regime, writing invented facts one at a time into a Qwen3 model already modified by previous writes, and varying the training data, method, and parameter update. Across SFT and off- and on-policy distillation, using LoRA or full fine-tuning, we compare repeated statement training (the same fact repeated in two formats) with varied example training (24 factual restatements) and find that varied examples comprehensively support more flexible use. After twenty sequential writes and merges, the model answers only 1% of questions about earlier facts correctly when every write uses repeated statements, compared with 46% when every write uses varied examples. We additionally show that this retention depends on the data used for the later writes, regardless of training method or parameter update, and that behavioral forgetting of an earlier fact does not erase its presence from the log-probabilities. Together, our framework neatly provides a comparison of performance across training data, training regimes and parameter update schemes in an iterative learning task.

1 Introduction

If this field can have an age-old question, then it is whether a model can learn som without forgetting what it already knows. Gaining such an understanding is critical to h architectures that are able to *learn* outside of just their provided context. One way question concrete is to examine how current language models and associated post-traini learn and remember new pieces of information in their weights. Singular facts offer a test of this ability, given that the ability to continually write self-contained pieces of k weights is the simplest such form of learning [Jang et al., 2022].

O’Neill et al.
“Can a Language Model Learn Facts Continually in Its Weights?”

Surprising result: Augmenting the examples for the next ten facts improved retention of the first by 38 percentage points!





Can a Language Model Continually Learn Facts in its Weights?

Charles O'Neill
Base Labs
charles.oneill@baseten.co

Max Kirkby
Base Labs
max.kirkby@baseten.co

Jonathon Liu
Base Labs
jonathon.liu@baseten.co

Michael Psenka
Base Labs
psenka@baseten.co

Abstract

Continual learning is a long-standing capability gap between LLMs and humans. Writing new knowledge into a model's weights routinely causes it to forget old knowledge, commonly denoted as "catastrophic forgetting". Various modifications of supervised fine-tuning and distillation aim to mitigate catastrophic forgetting, but quantifying what (or how much) information was forgotten is often difficult. In this paper, we study whether current methods of writing knowledge into weights enable models to learn continually without forgetting. We introduce a framework for studying continual learning in the iterative regime, writing invented facts one at a time into a Qwen3 model already modified by previous writes, and varying the training data, method, and parameter update. Across SFT and off- and on-policy distillation, using LoRA or full fine-tuning, we compare repeated statement training (the same fact repeated in two formats) with varied example training (24 factual restatements) and find that varied examples comprehensively support more flexible use. After twenty sequential writes and merges, the model answers only 1% of questions about earlier facts correctly when every write uses repeated statements, compared with 46% when every write uses varied examples. We additionally show that this retention depends on the data used for the later writes, regardless of training method or parameter update, and that behavioral forgetting of an earlier fact does not erase its presence from the log-probabilities. Together, our framework neatly provides a comparison of performance across training data, training regimes and parameter update schemes in an iterative learning task.

1 Introduction

If this field can have an age-old question, then it is whether a model can learn som without forgetting what it already knows. Gaining such an understanding is critical to h architectures that are able to *learn* outside of just their provided context. One way question concrete is to examine how current language models and associated post-traini learn and remember new pieces of information in their weights. Singular facts offer a test of this ability, given that the ability to continually write self-contained pieces of k weights is the simplest such form of learning [Jang et al., 2022].

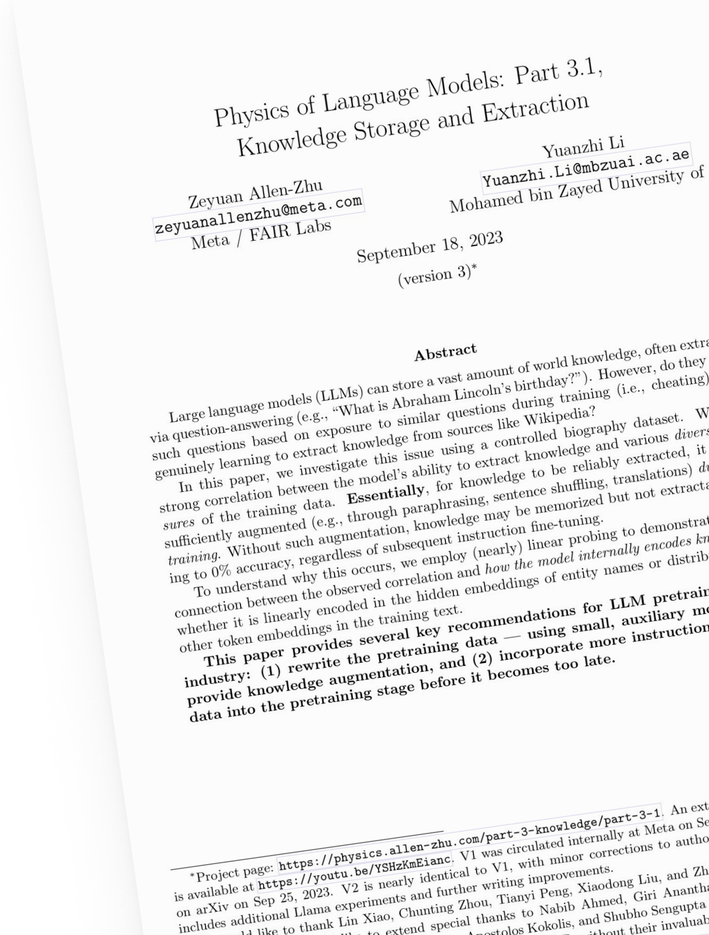
O'Neill et al.
"Can a Language Model Learn Facts Continually in Its Weights?"

Some related lessons from another study that observes a similar story:

> “for knowledge to be reliably extracted, it must be sufficiently augmented”

> “We emphasize the importance of pre-training data rewriting (augmentation), particularly for rare but critical data.”

Allen-Zhu, Li
“Physics of Language Models: Part 3.1, Knowledge Storage and Extraction”



Physics of Language Models: Part 3.1,
Knowledge Storage and Extraction

Zeyuan Allen-Zhu
zeyuanallenzhu@meta.com
Meta / FAIR Labs

Yuanzhi Li
Yuanzhi.Li@mbzuai.ac.ae
Mohamed bin Zayed University of

September 18, 2023
(version 3)*

Abstract

Large language models (LLMs) can store a vast amount of world knowledge, often extra
via question-answering (e.g., “What is Abraham Lincoln’s birthday?”). However, do they
such questions based on exposure to similar questions during training (i.e., cheating)
genuinely learning to extract knowledge from sources like Wikipedia?

In this paper, we investigate this issue using a controlled biography dataset. W
strong correlation between the model’s ability to extract knowledge and various *divers*
sures of the training data. **Essentially**, for knowledge to be reliably extracted, it
sufficiently augmented (e.g., through paraphrasing, sentence shuffling, translations) *d*
training. Without such augmentation, knowledge may be memorized but not extracta
ing to 0% accuracy, regardless of subsequent instruction fine-tuning.

To understand why this occurs, we employ (nearly) linear probing to demonstra
connection between the observed correlation and *how the model internally encodes kn*
whether it is linearly encoded in the hidden embeddings of entity names or distrib
other token embeddings in the training text.

This paper provides several key recommendations for LLM pretrain
industry: (1) rewrite the pretraining data — using small, auxiliary m
provide knowledge augmentation, and (2) incorporate more instruction
data into the pretraining stage before it becomes too late.

*Project page: https://physics.allen-zhu.com/part-3-knowledge/part-3-1. An ext
is available at https://youtu.be/YSHzKmEianc. V1 was circulated internally at Meta on Se
on arXiv on Sep 25, 2023. V2 is nearly identical to V1, with minor corrections to autho
includes additional Llama experiments and further writing improvements.
d like to thank Lin Xiao, Chunting Zhou, Tianyi Peng, Xiaodong Liu, and Zh
to extend special thanks to Nabib Ahmed, Giri Anantha
postolos Kokolis, and Shubho Sengupta
ithout their invaluab

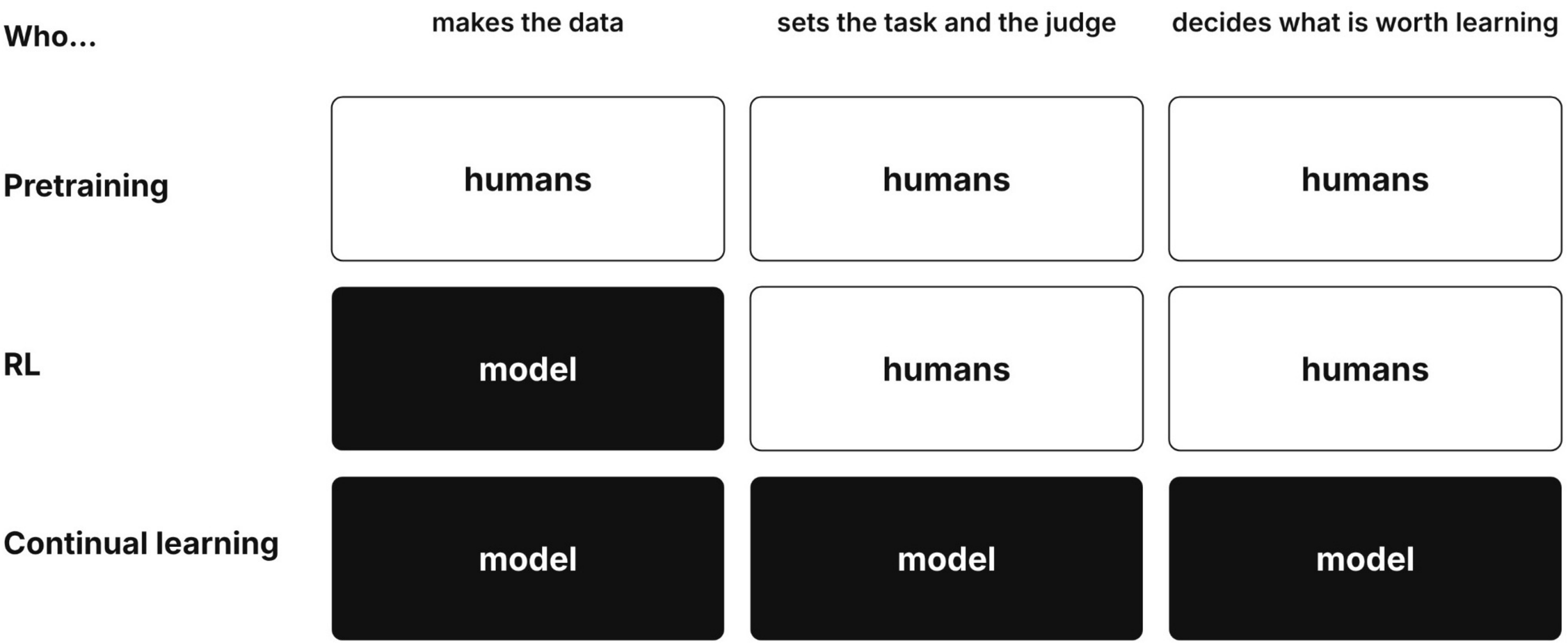
Pretraining does all of that already:

Consider a random fact about the world.
The data-generation process of the world (later fed as standard pretraining data) is such that...

1. Facts show up with augmentations. The world rewrites every fact in many forms and works out its consequences.
2. The frequency/proportion of a fact is naturally weighted by its importance.
3. Everything is shuffled together. Pretraining is i.i.d. Every batch is a random sample of everything and thus no single update overwrites what the rest of the data still needs. ("Catastrophic forgetting" is minimal)

A model at deployment does not have these privileges. It sees new facts sparsely, unaugmented, and rarely in a stream that's shuffled.

Related: Generalisation needs pressure: with spare capacity a model can memorise. With more data than capacity it's pushed to learn higher-order features.



after Beren Millidge, “Three Phases of Capabilities and Alignment” (2026)

Elephant in the room:

In-context learning.
Putting Zorvathine in the prompt results in the model getting the derived questions correct: 4% without the fact → 91% with it (lenient grading).

In context, the model needs no augmentation, as it reasons with the fact on the spot, which itself is perhaps pretraining's most important outcome.

ICL is strong proof of existence for models being able to "learn" and "use" sparse, unaugmented, non-i.i.d. facts non-trivially. Though this is obviously limited by the context lengths.



On the generalization of language models from in-context learning and finetuning: a controlled study

Andrew K. Lampinen*
Google DeepMind
Moutain View, CA
lampinen@google.com

Arslan Chaudry*
Google DeepMind
Moutain View, CA
arslanch@google.com

Stephanie C. Y. Chan*
Google DeepMind
San Francisco, CA
scychan@google.com

Cody Wild
Google DeepMind
Moutain View, CA

Diane Wan
Google DeepMind
Seattle, WA

Alex Ku
Google DeepMind
Mountain View, CA

Jörg Bornschein
Google DeepMind
London, UK

Razvan Pascanu
Google DeepMind
London, UK

Murray Shanahan
Google DeepMind
London, UK

James L. McClelland
Google DeepMind & Stanford University
Mountain View, CA

Abstract

Large language models exhibit exciting capabilities, yet can show surprisingly narrow generalization[2] from finetuning. E.g. they can fail to generalize to simple reversals of relations they are trained on, or fail to make simple logical deductions based on trained information. These failures to generalize factual information from fine-tuning can significantly hinder the reasoning capabilities of these models. On the other hand, language models' in-context learning shows different inductive biases and deductive reasoning capabilities. Here, we explore these differences in generalization and deductive reasoning between in-context- and fine-tuning-based learning. To do so, we constructed several novel datasets to evaluate and improve models' abilities to make generalizations over factual information from novel data. These datasets are designed to create clean tests of generalization, by isolating the knowledge in the dataset from that in pretraining. We expose pretrained large models to controlled subsets of the information in these datasets — either in context, or through fine-tuning — and evaluate their performance on test sets that require various types of generalization. We find overall that in data-matched settings, in-context learning can generalize several types of inferences more flexibly than fine-tuning (though we also find some qualifications of prior findings such as cases when fine-tuning can generalize to reversals embedded in a larger structure of knowledge). We build on these findings to propose a method to enable improved generalization from fine-tuning: adding in-context reasoning traces to finetuning data. We show that this method improves generalization and deductive reasoning across various splits of our datasets and other benchmarks. Our results have implications for understanding the different forms of reasoning afforded by different modes of learning in language models, and practically improving performance.

*Equal contribution.

O'Neill et al.
"Can a Language Model Learn Facts Continually in Its Weights?"

“We find overall that in data-matched settings, in-context learning can generalize several types of inferences more flexibly than fine-tuning”

“We build on these findings to propose a method to enable improved generalization from fine-tuning: adding in-context reasoning traces to finetuning data. We show that this method improves generalization and deductive reasoning across various splits of our datasets and other benchmarks”

This is intuitively verified in practice by the current barrage of coding agent harnesses. Reasoning helps as well.

Lampinen et al.
“On the generalization of language models from in-context learning and finetuning: a controlled study”



On the generalization of language models from in-context learning and finetuning: a controlled study

Andrew K. Lampinen* Google DeepMind Moutain View, CA lampinen@google.com

Arslan Chaudry* Google DeepMind Moutain View, CA arslanch@google.com

Stephanie C. Y. Chan* Google DeepMind San Francisco, CA scychan@google.com

Cody Wild Google DeepMind Moutain View, CA

Diane Wan Google DeepMind Seattle, WA

Alex Ku Google DeepMind Mountain View, CA

Jörg Bornschein Google DeepMind London, UK

Razvan Pascanu Google DeepMind London, UK

Murray Shanahan Google DeepMind London, UK

James L. McClelland Google DeepMind & Stanford University Mountain View, CA

Abstract

Large language models exhibit exciting capabilities, yet can show surprisingly narrow generalization[2] from finetuning. E.g. they can fail to generalize to simple reversals of relations they are trained on, or fail to make simple logical deductions based on trained information. These failures to generalize factual information from fine-tuning can significantly hinder the reasoning capabilities of these models. On the other hand, language models' in-context learning shows different inductive biases and deductive reasoning capabilities. Here, we explore these differences in generalization and deductive reasoning between in-context- and fine-tuning-based learning. To do so, we constructed several novel datasets to evaluate and improve models' abilities to make generalizations over factual information from novel data. These datasets are designed to create clean tests of generalization, by isolating the knowledge in the dataset from that in pretraining. We expose pretrained large models to controlled subsets of the information in these datasets — either in context, or through fine-tuning — and evaluate their performance on test sets that require various types of generalization. We find overall that in data-matched settings, in-context learning can generalize several types of inferences more flexibly than fine-tuning (though we also find some qualifications of prior findings such as cases when fine-tuning can generalize to reversals embedded in a larger structure of knowledge). We build on these findings to propose a method to enhance improved generalization from fine-tuning: adding in-context reasoning traces to finetuning data. We show that this method improves generalization and deductive reasoning across various splits of our datasets and other benchmarks. Our results have implications for understanding the different forms of reasoning afforded by different modes of learning in language models, and practically improving their performance.

*Equal contribution.

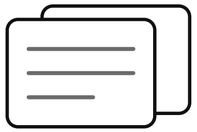


Context

Writing
instant: just read it

Lasts
this session only

Catch
costs memory for
every token it holds, hard limits

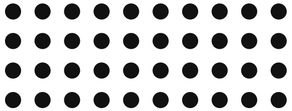


External notes

Writing
cheap: write a file

Lasts
until deleted

Catch
only as good as what
got written down



Weights

Writing
slow: gradient steps

Lasts
permanently

Catch
shared by everything it knows:
new writes disturb old ones

fast and temporary

slow and lasting

ICL is not free:

For Llama 3.1 70B (80 layers, 8 KV heads, h_dim = 128, 16-bit precision), size is 320 KiB per token. At roughly 430k tokens the cache is larger than the model's own weights.

But, externalisation in many forms is rampant already. memory.md files, external knowledge stores, trained context window compaction, cartridges (more later!) Longer context windows cannot be ruled out either.

Eyuboglu, Ehrlich, Arora et al.
"Cartridges: Lightweight and general-purpose long context representations via self-study"

Restating the criteria:

1. Correctly choose what is to be learnt, and learn it at the right level of abstraction with as few samples as possible
2. Don’t overwrite what the base model knew
3. Don’t forget what the earlier updates added
4. Don’t degrade general capability, including the potential of learning the next thing.

Quick rundown of current methods that address related issues:
Tradeoffs between: learns from one exposure · scales without degrading · uses what it learned · **decides what matters**

Train on your own samples

On-policy RL, on-policy distillation, self-distillation (SDFT, OPSD, SDPO).

Don’t write to weights at all

Cartridges, Still, KV externalisation, iterative compaction, context language models.

“Dreaming”

Self-play-like data: rewrites, replay, simulations, automated env creation.

Context length unlocks

Better architectures (linear attention, hybrids).

Targeted updates/Memory-first architectural components

Lin et al. sparse memory finetuning, DeepSeek Engram modules, test-time training (Titans, TTT-E2E).

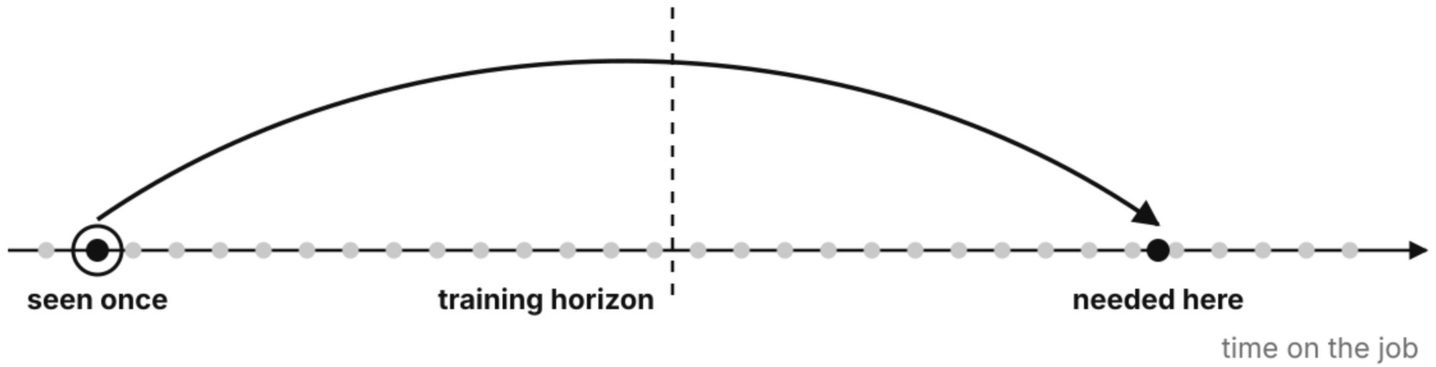
Model updates itself

SEAL (Zweiger et al.), weight-space programming (Takehi et al.)

On Compaction:

Compaction clearly works! Models trained through it learn what to keep, and this is optimised in a closed loop instead of just a basic “summarise” prompt. Ideas like CLMs go further: they edit their own context directly.

But these are learned transformations (feeding on backpropagated signals from RL rewards in designed envs). They learn what matters only for payoffs inside their training horizon.



RL/train on your own samples

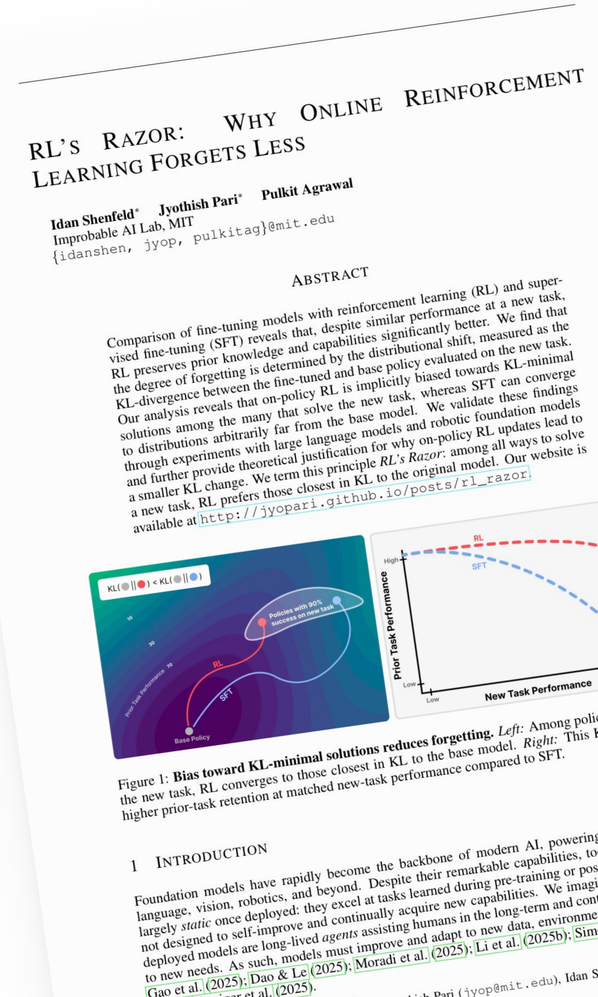
For a task/completion of interest...

SFT: take this gold-standard sample and memorise it.
This memorises something EXTERNAL.

On-policy RL: sample some completions, a reward picks the best ones.
This upweights something INTERNAL.

Moving away from the base model is bad (measured in KL divergence). For matched accuracy, RL is shown to have less KL div than SFT.

Shenfeld, Pari, Agrawal
"RL's Razor: Why Online Reinforcement Learning Forgets Less"



RL'S RAZOR: WHY ONLINE REINFORCEMENT LEARNING FORGETS LESS

Idan Shenfeld* **Jyothish Pari*** **Pulkit Agrawal**
Improbable AI Lab, MIT
{idanshen, jyop, pulkitag}@mit.edu

ABSTRACT

Comparison of fine-tuning models with reinforcement learning (RL) and supervised fine-tuning (SFT) reveals that, despite similar performance at a new task, RL preserves prior knowledge and capabilities significantly better. We find that the degree of forgetting is determined by the distributional shift, measured as the KL-divergence between the fine-tuned and base policy evaluated on the new task. Our analysis reveals that on-policy RL is implicitly biased towards KL-minimal solutions among the many that solve the new task, whereas SFT can converge to distributions arbitrarily far from the base model. We validate these findings through experiments with large language models and robotic foundation models and further provide theoretical justification for why on-policy RL updates lead to a smaller KL change. We term this principle *RL's Razor*: among all ways to solve a new task, RL prefers those closest in KL to the original model. Our website is available at http://jyopari.github.io/posts/rl_razor

Figure 1: **Bias toward KL-minimal solutions reduces forgetting.** *Left:* Among policies the new task, RL converges to those closest in KL to the base model. *Right:* This KL higher prior-task retention at matched new-task performance compared to SFT.

1 INTRODUCTION

Foundation models have rapidly become the backbone of modern AI, powering ap language, vision, robotics, and beyond. Despite their remarkable capabilities, today largely *static* once deployed: they excel at tasks learned during pre-training or post-tra not designed to self-improve and continually acquire new capabilities. We imagine deployed models are long-lived *agents* assisting humans in the long-term and continu to new needs. As such, models must improve and adapt to new data, environments, Gao et al. (2025); Dao & Le (2025); Moradi et al. (2025); Li et al. (2025b); Simond

et al. (2025).
Pari (jyop@mit.edu), Idan Shen

Self-distillation methods.

On-policy RL assumes your model can produce an acceptable response. In many cases when trying to learn something new, it cannot.

On-policy distillation: The student still samples its own answers, but a teacher scores every token ("I'd have put more probability here"). Dense feedback, but still assumes the existence of a stronger teacher.

Self-distillation (SDFT, OPSD, SDPO): in absence of a stronger teacher, use the same model, with the answer in its context as a hint. That hint could be a demonstration (SDFT), a worked solution (OPSD), or the error message from its own failed attempt (SDPO).

Intuitively: you use in-context learning to bootstrap a teacher, and get a low KL weight update.

Lu et al. "On-Policy Distillation" · Shenfeld et al. "Self-Distillation Enables Continual Learning"
Zhao et al. "Self-Distilled Reasoner: On-Policy Self-Distillation for Large Language Models"
Hübotter et al. "Reinforcement Learning via Self-Distillation"

Issues:

Bottlenecked by in-context learning ability, does not make “learning what to learn” easier, RL can learn atomic behaviors without learning the underlying knowledge.

Practitioners aren’t very positive w.r.t. OPSD!



takeaway: super strong low-forgetting, low-interference method
IF DONE CORRECTLY. Definitely better than SFT.

“Don’t write to weights at all”

Cartridges:
Train a small KV cache on a corpus by “self-study”. It matches full in-context quality with 38× less memory.

A trained KV cache is approximable to an MLP layer[1]. So a bank of cartridges plus a router that picks one is taking MoEs to the limit: one expert per context.

Again, exploits ICL as a fast-learner.



Cartridges: Lightweight and general-purpose long context representations via self-study

Sabri Eyuboglu [1*] Ryan Ehrlich [1*] Simran Arora [1,2*] Neel Guha [1] Dylan Zinsley [3] Em
Will Tennien [1] Atri Rudra [3] James Zou [1] Azalia Mirhoseini [1] Christopher Ré [1]

[1]Stanford University [2] Caltech [3]University at Buffalo * Equal contribution

eyuboglu@stanford.edu, rehrlich@stanford.edu, simarora@stanford.edu

HazyResearch/cartridges

Abstract

Large language models are often used to answer queries grounded in large text corpora (e.g. code
documents, or chat histories) by placing the entire corpus in the context window and leveraging
learning (ICL). Although current models support contexts of 100K–1M tokens, this setup is cos
because the memory consumption of the KV cache scales with input length. We explore an
training a smaller KV cache offline on each corpus. At inference time, we load this trained KV-
we call a CARTRIDGE, and decode a response. Critically, the cost of training a CARTRIDGE can
across all the queries referencing the same corpus. However, we find that the naive approac
the CARTRIDGE with next-token prediction on the corpus is not competitive with ICL. Instead
SELF-STUDY, a training recipe in which we generate synthetic conversations about the co
the CARTRIDGE with a context-distillation objective. We find that CARTRIDGES trained with
replicate the functionality of ICL, while being significantly cheaper to serve. On challenging
benchmarks, CARTRIDGES trained with SELF-STUDY match ICL performance while using 38.6
and enabling 26.4× higher throughput. SELF-STUDY also extends the model’s effective
(e.g. from 128k to 484k tokens on MTOB) and surprisingly, leads to CARTRIDGES that can
inference time without retraining.

1 Introduction

Large language model (LLM) users often place large text corpora into the context window.
or organization may use LLMs to understand codebases [63], financial documents [38],
textbooks [68], or personal files [7]. LLMs excel here due to in-context learning (ICL)
responses to diverse queries (e.g., factual Q&A, summarization, code generation) [24].

Despite its flexibility, this usage paradigm is costly to serve. ICL requires maintaining a
linearly with the input length. For example, LLaMA 70B needs 84 GB of memory (a
answer a single question over a 128k-token context [25]. This severely limits user th
H100 GPU, LLaMA 8B’s peak throughput (tokens/s) drops by 77× when increasing
120k tokens (Figure 3).

Prior work has thus explored ways to reduce KV cache memory usage. For insta
sion methods reduce the number of tokens stored in the cache using summarizati
filtering [21, 42, 55], while KV cache compression techniques directly compres
pairs [27, 67, 84, 114]. Unfortunately, there are memory-quality tradeoffs associated
on challenging long-context tasks, we find that performance degrade
ratios greater than 2× (see Figure 4).

Eyuboglu, Ehrlich, Arora et al.
“Cartridges: Lightweight and general-purpose long context representations via self-study”
[1] Sukhbaatar et al. “Augmenting Self-attention with Persistent Memory”

Dreaming/self-play

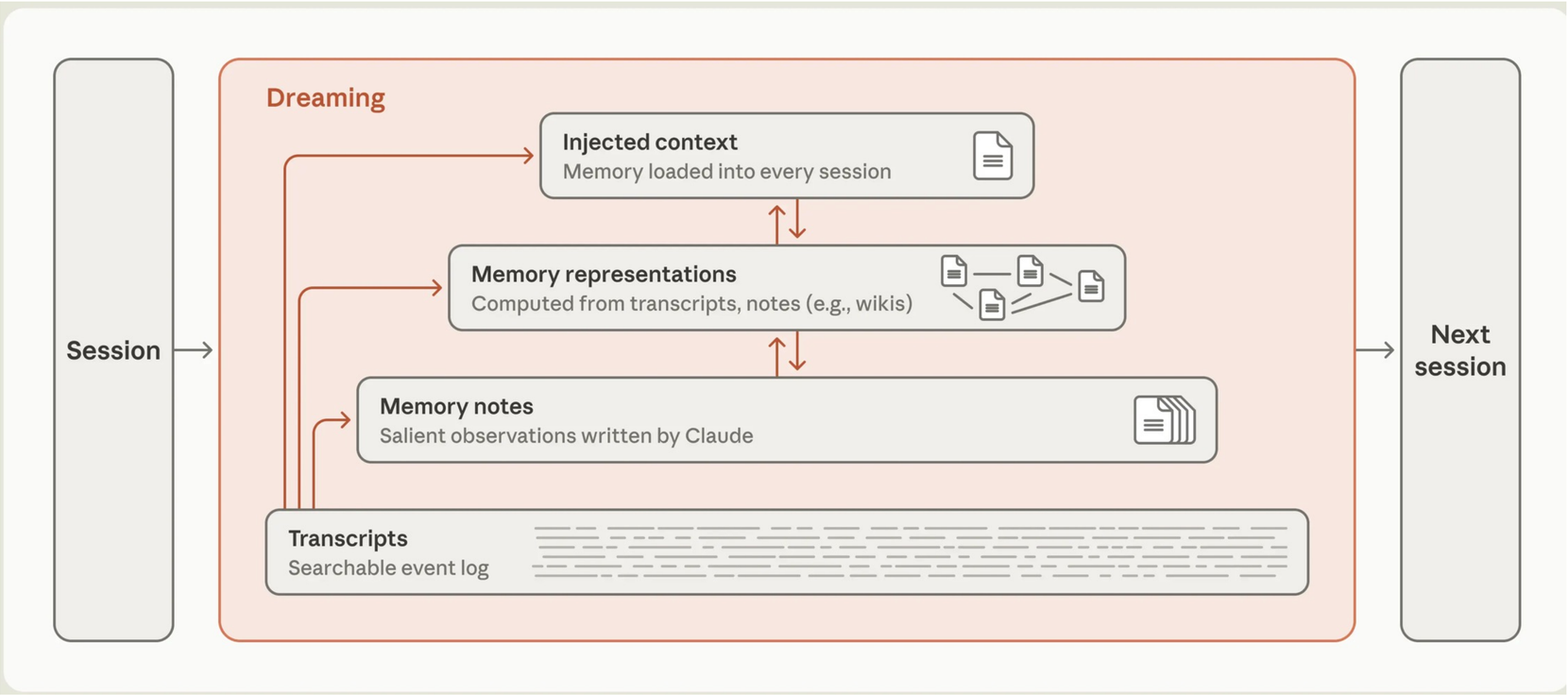
Claude Code Auto Dream: Between sessions, a background agent merges, fixes and prunes its own memory files. (leaked feature)

- **During a session:** the model writes notes using its best guess about what's salient.
- **Between sessions:** it goes back over the record with hindsight and reorganises. That offline pass is what they call **dreaming**.

scope: defines memory as context saved between sessions, and explicitly excludes changes to the weights.

Bonus tailwind: the advent of user simulator models.

Lance Martin, Anthropic
"The components of agent memory"



“Claude has learned to judge salience”
On Pokémon Red: “Sonnet 3.7 assumed everything might matter and wrote to memory indiscriminately [...] and earned one badge. [...] Fable 5, by comparison, wrote nine files in two folders, plus a single top-level file called `global.md`, and earned four badges”

Lance Martin, Anthropic
“The components of agent memory”

The model learns to update itself:

SEAL: Given a passage, instead of finetuning on the passage itself, model writes its own training data about it, then finetunes on that.

generate several self-edits, finetune a copy of the model on each, and test the result. Reward = 1 if the update improved performance, 0 if not. Keep the edits that worked and train the model to write more like them

pros: direct data augmentation, model learns how to update itself.
cons: still has to be given what to learn! reward is super short-horizon i.e. does it help on downstream questions (mostly just knowledge tasks)



Self-Adapting Language Models

Adam Zweiger* Jyothish Pari*† Han Guo Ekin Akyürek Yoon Kim Pulkit Agrawa

Massachusetts Institute of Technology

{adamz, jyop, hanguo, akyurek, yoonkim, pulkitag}@mit.edu

Abstract

Large language models (LLMs) are powerful but static; they lack mechanisms to adapt their weights in response to new tasks, knowledge, or examples. We introduce Self-Adapting LLMs (SEAL), a framework that enables LLMs to self-adapt by generating their own finetuning data and update directives. Given a new input, the model produces a *self-edit*—a generation that may restructure the information in different ways, specify optimization hyperparameters, or invoke tools for data augmentation and gradient-based updates. Through supervised finetuning (SFT), these self-edits result in persistent weight updates, enabling lasting adaptation. To train the model to produce effective self-edits, we use a reinforcement learning loop, using the downstream performance of the updated model as the reward signal. Unlike prior approaches that rely on separate adaptation modules or auxiliary networks, SEAL directly uses the model's generation to parameterize and control its own adaptation process. Experiments on knowledge incorporation and few-shot generalization show that SEAL is a promising step toward language models capable of self-directed adaptation in response to new data. Our website and code is available at https://jyopari.github.io/posts/seal.

1 Introduction

Large language models (LLMs) pretrained on vast text corpora exhibit remarkable abilities in understanding and generation [1, 2, 3, 4, 5]. However, adapting these powerful models fo tasks [6], integrating new information [7], or mastering novel reasoning skills [8] remains c due to the limited availability of task-specific data. In this paper, we explore an intriguing can an LLM self-adapt by transforming or generating its own training data and learning

As an analogy, consider a human student preparing for the final exam of a machine lea Many students rely on their notes to prepare for the exam. These notes are often deriv lecture content, textbooks, or information available on the internet. Instead of relying students to understand the content and answer exam questions. This phenomenon of r and augmenting external knowledge in a way that is easier to understand is not limited exams, but seems to be universally true of human learning across tasks. Furtherm humans assimilate information in different ways—some might condense the informatio diagram, some into text, or some might rely more on concrete mathematical descriptio

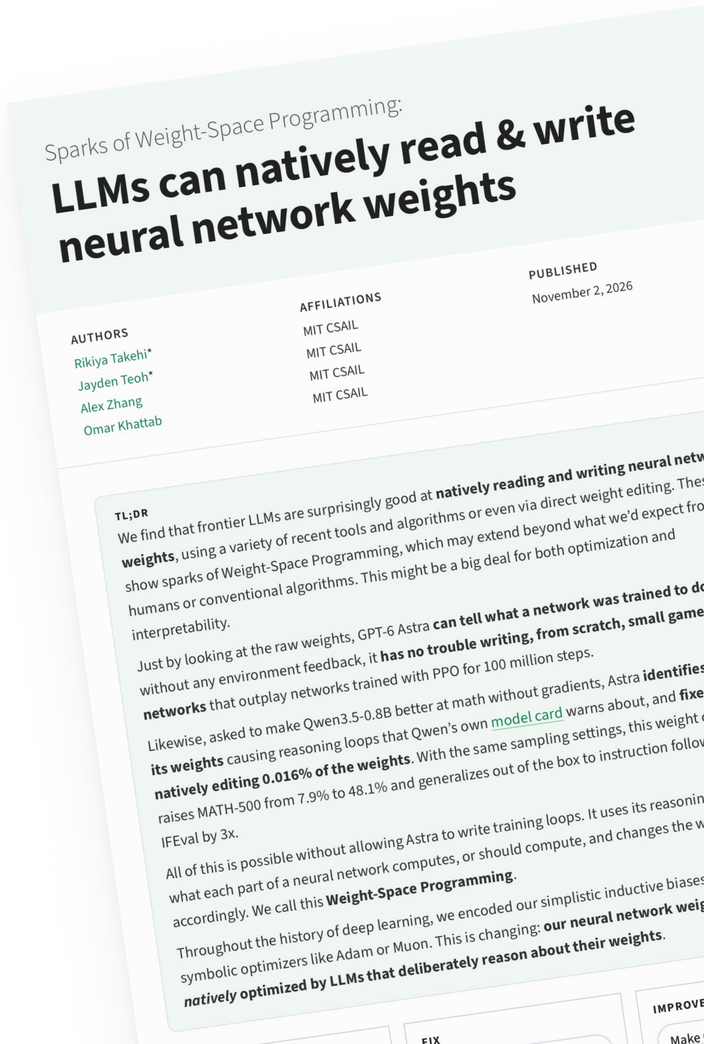
Such assimilation, restructuring, or rewriting of data as part of the learning proces with how large language models (LLMs) are typically trained and deployed. Gi current LLMs consume and learn from the task data "as-is" via finetuning or in-c [9, 10, 11, 12]. However, such data may not be in an optimal format (or volume) f

Zweiger, Pari et al.
"Self-Adapting Language Models"

The model learns to update itself, part 2:

GPT-6 Astra, when given Qwen3.5-0.8B's raw weights and a budget of forward passes, with no gradients, edits 0.016% of them. MATH-500: 7.9% → 48.1%.

Super early ⚠: it removed a fault hiding an ability, it couldn't add one. But a taste for things to come.


Sparks of Weight-Space Programming:

LLMs can natively read & write neural network weights

AUTHORS

Rikiya Takehi*
Jayden Teoh*
Alex Zhang
Omar Khattab

AFFILIATIONS

MIT CSAIL
MIT CSAIL
MIT CSAIL
MIT CSAIL

PUBLISHED

November 2, 2026

TL;DR

We find that frontier LLMs are surprisingly good at **natively reading and writing neural netw**
weights, using a variety of recent tools and algorithms or even via direct weight editing. The
show sparks of Weight-Space Programming, which may extend beyond what we'd expect fro
humans or conventional algorithms. This might be a big deal for both optimization and
interpretability.

Just by looking at the raw weights, GPT-6 Astra **can tell what a network was trained to d**
without any environment feedback, it **has no trouble writing, from scratch, small game**
networks that outplay networks trained with PPO for 100 million steps.

Likewise, asked to make Qwen3.5-0.8B better at math without gradients, Astra **identifies**
its weights causing reasoning loops that Qwen's own model card warns about, and **fixe**
natively editing 0.016% of the weights. With the same sampling settings, this weight
raises MATH-500 from 7.9% to 48.1% and generalizes out of the box to instruction follo
IFEval by 3x.

All of this is possible without allowing Astra to write training loops. It uses its reasonin
what each part of a neural network computes, or should compute, and changes the w
accordingly. We call this **Weight-Space Programming**.

Throughout the history of deep learning, we encoded our simplistic inductive biases
symbolic optimizers like Adam or Muon. This is changing: **our neural network weig**
***natively* optimized by LLMs that deliberately reason about their weights**.

IMPROVE

FIX

Takehi et al.
"Sparks of Weight-Space Programming:
LLMs can natively read & write neural network weights"

The model learns to update itself, part 3:

“Context Language Models”

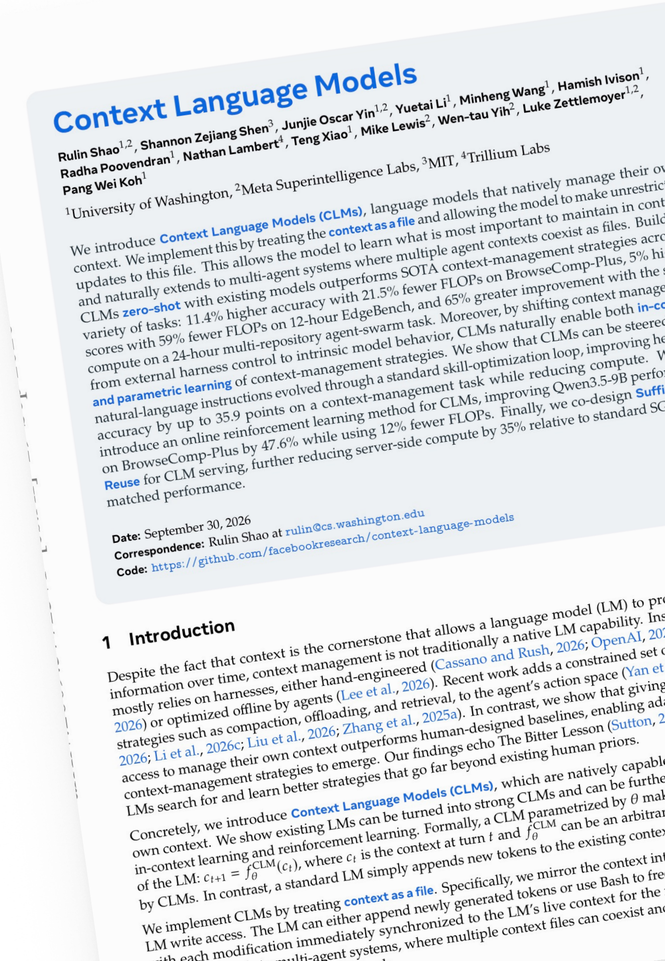
Allow models to edit their own context by making it a file with unrestricted edits.
Instead of appending forever or summarising at the end, the model edits its own context like a file: rewrite, delete, insert.

$$c_{t+1} = f_\theta^{\mathrm{CLM}}(c_t)$$

Against summary compaction (same model, 32K budget): TBLite 73.7% vs 67.0%; 12-hour EdgeBench runs 44.6 vs 42.3 at ~60% less compute.

Also worth checking out: RLMs

Shao et al. “Context Language Models”
Zhang, Kraska, Khattab “Recursive Language Models”


Context Language Models

Rulin Shao[1,2], Shannon Zejiang Shen[3], Junjie Oscar Yin[1,2], Yuetai Li[1], Minheng Wang[1], Hamish Ivison[1], Radha Poovendran[1], Nathan Lambert[4], Teng Xiao[1], Mike Lewis[2], Wen-tau Yih[2], Luke Zettlemoyer[1,2], Pang Wei Koh[1]

[1]University of Washington, [2]Meta Superintelligence Labs, [3]MIT, [4]Trillium Labs

We introduce Context Language Models (CLMs), language models that natively manage their ow context. We implement this by treating the context as a file and allowing the model to make unrestric updates to this file. This allows the model to learn what is most important to maintain in cont and naturally extends to multi-agent systems where multiple agent contexts coexist as files. Build CLMs zero-shot with existing models outperforms SOTA context-management strategies acro variety of tasks: 11.4% higher accuracy with 21.5% fewer FLOPs on BrowseComp-Plus, 5% hi scores with 59% fewer FLOPs on 12-hour EdgeBench, and 65% greater improvement with the s compute on a 24-hour multi-repository agent-swarm task. Moreover, by shifting context manage from external harness control to intrinsic model behavior, CLMs naturally enable both in-co and parametric learning of context-management strategies. We show that CLMs can be steered natural-language instructions evolved through a standard skill-optimization loop, improving he accuracy by up to 35.9 points on a context-management task while reducing compute. W introduce an online reinforcement learning method for CLMs, improving Qwen3.5-9B perfo on BrowseComp-Plus by 47.6% while using 12% fewer FLOPs. Finally, we co-design Suffi Reuse for CLM serving, further reducing server-side compute by 35% relative to standard SG matched performance.

Date: September 30, 2026
Correspondence: Rulin Shao at rulin@cs.washington.edu
Code: https://github.com/facebookresearch/context-language-models

1 Introduction

Despite the fact that context is the cornerstone that allows a language model (LM) to pr information over time, context management is not traditionally a native LM capability. Ins mostly relies on harnesses, either hand-engineered (Cassano and Rush, 2026; OpenAI, 20 2026) or optimized offline by agents (Lee et al., 2026). Recent work adds a constrained set strategies such as compaction, offloading, and retrieval, to the agent’s action space (Yan et 2026; Li et al., 2026c; Liu et al., 2026; Zhang et al., 2025a). In contrast, we show that giving access to manage their own context outperforms human-designed baselines, enabling ad context-management strategies to emerge. Our findings echo The Bitter Lesson (Sutton, 2 LMs search for and learn better strategies that go far beyond existing human priors.

Concretely, we introduce Context Language Models (CLMs), which are natively capable own context. We show existing LMs can be turned into strong CLMs and can be further in-context learning and reinforcement learning. Formally, a CLM parametrized by θ mak of the LM: $c_{t+1} = f_\theta^{\mathrm{CLM}}(c_t)$, where c_t is the context at turn t and f_θ^{CLM} can be an arbitra by CLMs. In contrast, a standard LM simply appends new tokens to the existing conte

We implement CLMs by treating context as a file. Specifically, we mirror the context int LM write access. The LM can either append newly generated tokens or use Bash to fre with each modification immediately synchronized to the LM’s live context for the multi-agent systems, where multiple context files can coexist and

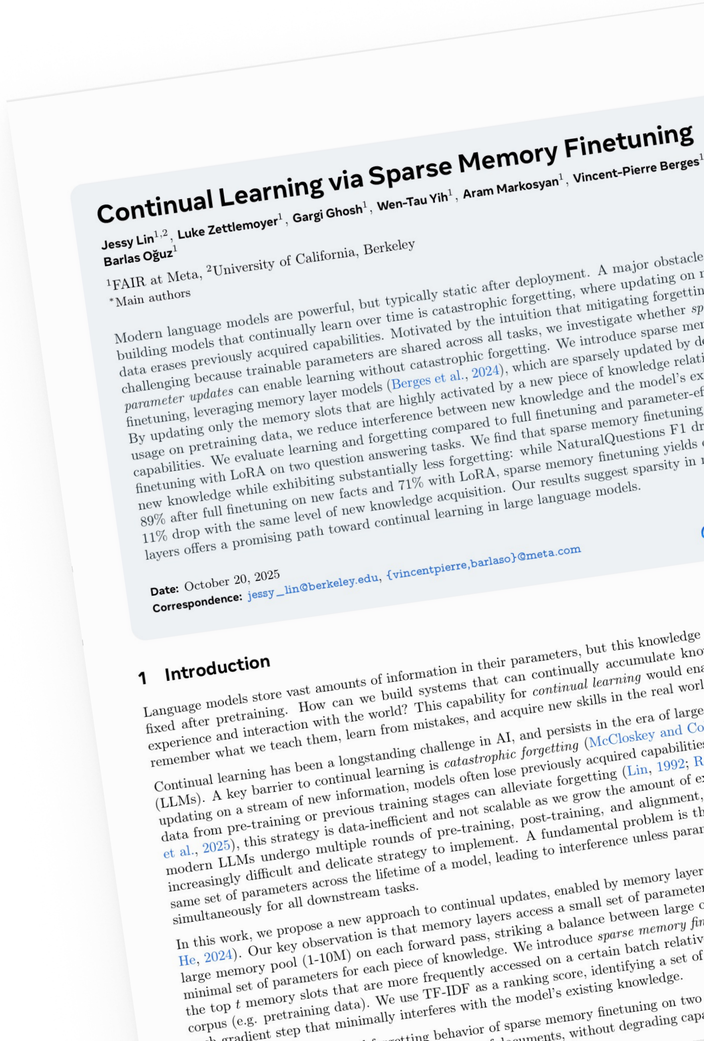
Architectural changes:

“Sparse Memory Finetuning”

A memory layer replaces some feedforward layers with a very large table of memory “slots”.

When learning something new, don’t update every slot the new data touches, use some metrics to target “slots” to edit.

High update sparsity, forgetting: 89% (full FT) → 11% (Memory slots)


Continual Learning via Sparse Memory Finetuning

Jessy Lin[1,2], Luke Zettlemoyer[1], Gargi Ghosh[1], Wen-Tau Yih[1], Aram Markosyan[1], Vincent-Pierre Berges[1]
Barlas Oğuz[1]

[1]FAIR at Meta, [2]University of California, Berkeley
*Main authors

Modern language models are powerful, but typically static after deployment. A major obstacle
building models that continually learn over time is catastrophic forgetting, where updating on
data erases previously acquired capabilities. Motivated by the intuition that mitigating forgettin
challenging because trainable parameters are shared across all tasks, we investigate whether sp
parameter updates can enable learning without catastrophic forgetting. We introduce sparse me
finetuning, leveraging memory layer models (Berges et al., 2024), which are sparsely updated by d
By updating only the memory slots that are highly activated by a new piece of knowledge relati
usage on pretraining data, we reduce interference between new knowledge and the model's ex
capabilities. We evaluate learning and forgetting compared to full finetuning and parameter-ef
finetuning with LoRA on two question answering tasks. We find that sparse memory finetuning
new knowledge while exhibiting substantially less forgetting: while NaturalQuestions F1 dr
89% after full finetuning on new facts and 71% with LoRA, sparse memory finetuning yields
11% drop with the same level of new knowledge acquisition. Our results suggest sparsity in
layers offers a promising path toward continual learning in large language models.

Date: October 20, 2025
Correspondence: jessy_lin@berkeley.edu, {vincentpierre,barlaso}@meta.com

1 Introduction

Language models store vast amounts of information in their parameters, but this knowledge
fixed after pretraining. How can we build systems that can continually accumulate kno
experience and interaction with the world? This capability for *continual learning* would ena
remember what we teach them, learn from mistakes, and acquire new skills in the real wor

Continual learning has been a longstanding challenge in AI, and persists in the era of large
(LLMs). A key barrier to continual learning is *catastrophic forgetting* (McCloskey and Co
updating on a stream of new information, models often lose previously acquired capabilities
data from pre-training or previous training stages can alleviate forgetting (Lin, 1992; R
et al., 2025), this strategy is data-inefficient and not scalable as we grow the amount of e
modern LLMs undergo multiple rounds of pre-training, post-training, and alignment,
increasingly difficult and delicate strategy to implement. A fundamental problem is th
same set of parameters across the lifetime of a model, leading to interference unless param
simultaneously for all downstream tasks.

In this work, we propose a new approach to continual updates, enabled by memory layer
He, 2024). Our key observation is that memory layers access a small set of parameter
large memory pool (1-10M) on each forward pass, striking a balance between large c
minimal set of parameters for each piece of knowledge. We introduce *sparse memory fi*
the top t memory slots that are more frequently accessed on a certain batch relativ
corpus (e.g. pretraining data). We use TF-IDF as a ranking score, identifying a set of
gradient step that minimally interferes with the model's existing knowledge.
forgetting behavior of sparse memory finetuning on two
uments, without degrading cap

Lin et al.
“Continual Learning via Sparse Memory Finetuning”

More architectural changes

DeepSeek Engram:
Add a huge table of embeddings indexed by hashing the last few tokens (an N-gram). The model's hidden state decides how much of the retrieved vector to use (a gate). i.e. learns when to look things up vs. compute them.

New architectural changes to scale up context length, ex:
Zyphra Hierarchical Attention, a sparse attention method that constructs and searches through a hierarchical tree to determine the relevant set of tokens for each query.


Subquadratic Attention with a Hierarchical Tree-Structured Indexer

Leon Lufkin*, Cutter Dawes*, Alexander Speicher, Kamesh Krishnamurthy, Tomás Figliolia, Beren Millidge

[1] Zyphra Research *Equal contribution

Correspondence: {leon,kamesh,tom}@zyphra.com

deepseek

Conditional Memory via Scalable Lookup:
A New Axis of Sparsity for Large Language Models

Xin Cheng[1,2*], Rui Tian[2*], Wangding Zeng[2], Damai Dai[2], Qinyu Chen[2],
Bingxuan Wang[2], Zhenda Xie[2], Kezhao Huang[2], Xingkai Yu[2]
Chengqi Deng[2], Shangyan Zhou[2], Chenggang Zhao[2], Zhewen Hao[2]
Yukun Li[2], Han Zhang[2], Zhengyan Zhang[2], Yixuan Wei[2], M.Y Xu[2]
Huishuai Zhang[1], Dongyan Zhao[1], Wenfeng Liang[2]
[1]Peking University [2]DeepSeek-AI
{zhanghuishuai, zhaody}@pku.edu.cn
{chengxin, tianr22, zengwangding, damai.dai}@deepseek.com

Abstract

While Mixture-of-Experts (MoE) scales capacity via conditional computation, Transformers lack a native primitive for knowledge lookup, forcing them to inefficiently simulate retrieval through computation. To address this, we introduce conditional memory as a complementary sparsity axis, instantiated via **Engram**, a module that modernizes classic N-gram embedding for $O(1)$ lookup. By formulating the *Sparsity Allocation* problem, we uncover a U-shaped scaling law that optimizes the trade-off between neural computation (MoE) and static memory (Engram). Guided by this law, we scale Engram to 27B parameters, achieving superior performance over a strictly iso-parameter and iso-FLOPs MoE baseline. Most notably, while the memory module is expected to aid knowledge retrieval (e.g., MMLU +3.4; CMMLU +4.0), we observe even larger gains in general reasoning (e.g., BBH +5.0; ARC-Challenge +3.7) and code/math domains (HumanEval +3.0; MATH +2.4). Mechanistic analyses reveal that Engram relieves the backbone's early layers from static reconstruction, effectively deepening the network for complex reasoning. Furthermore, by delegating local dependencies to lookups, it frees up attention capacity for global context, substantially boosting long-context retrieval (e.g., Multi-Query NIAH: 84.2 → 97.0). Finally, Engram establishes infrastructure-aware efficiency: its deterministic addressing enables runtime prefetching from host memory, incurring negligible overhead. We envision conditional memory as an indispensable modeling primitive for next-generation sparse models. Code available at: https://github.com/deepseek-ai/Engram.

1. Introduction

Sparsity is a recurring design principle for intelligent systems, spanning from biological neural circuits (Lennie, 2003; Olshausen and Field, 1997) to modern Large Language Models (LLMs). Currently, this principle is primarily realized through Mixture-of-Experts (MoE) (Dai et al., 2024; Shazeer et al., 2017), which

arXiv:2601.07372v2 [cs.CL] 12 Jul 2026

Cheng et al. "Conditional Memory via Scalable Lookup:
A New Axis of Sparsity for Large Language Models"
Lufkin et al. "Subquadratic Attention with a Hierarchical Tree-Structured Indexer"

Rough classification:

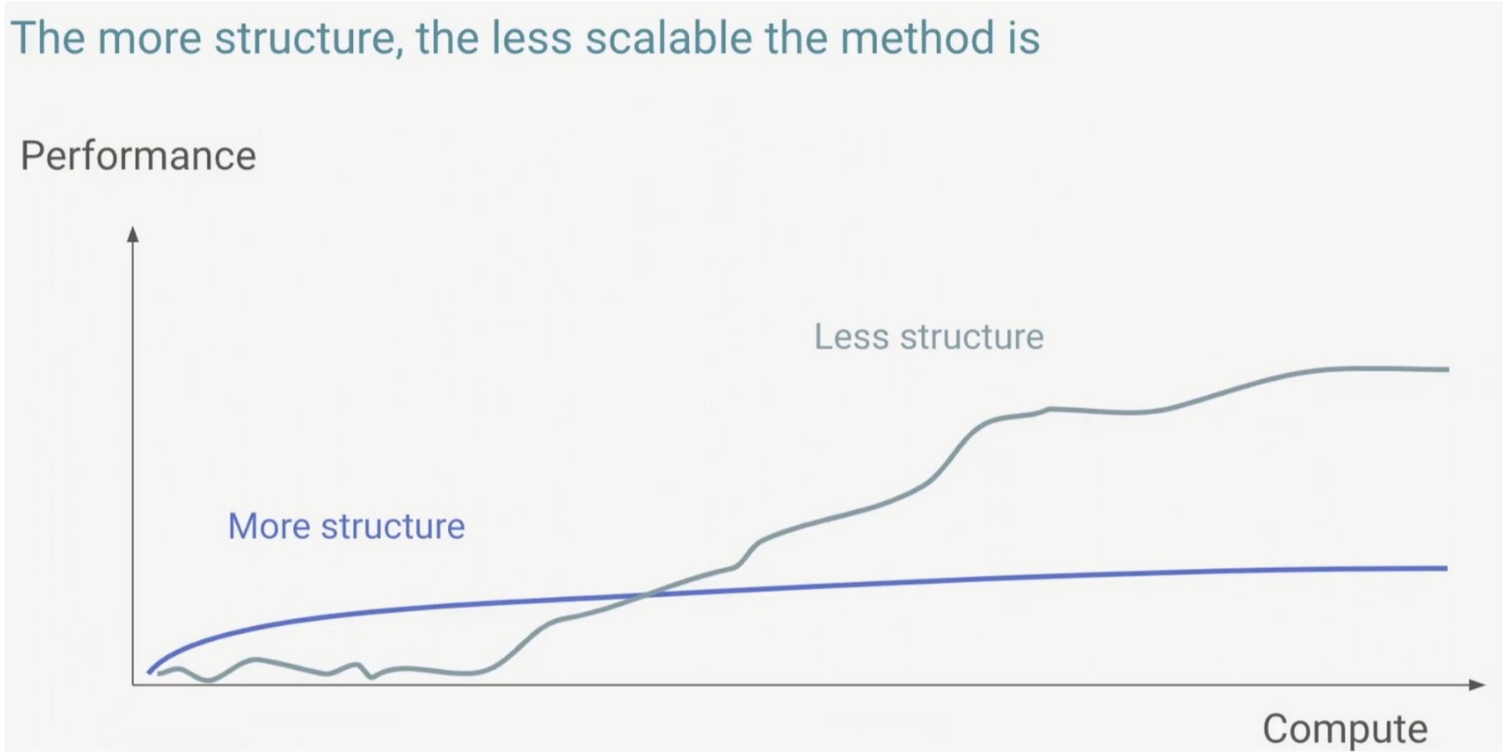
	Persists past the session	Learns from one exposure	Scales without degrading	Uses what it learned	Decides what matters
Long context / ICL	👎	👍	👎	👍	➖
Compaction, cartridges, memory files	🤷	👍	🤷	🤷	🤷
Plain finetuning (SFT)	👍	🤷	👎	👎	👎
Dreaming (rewrites, replay)	👍	👍	🤔	👍	🤷
Train on own samples (RL, self-distillation)	👍	🤷	🤷	👍	👎
Memory-first architectures	👍	🤷	👍	🤔	🤷
Model updates itself (SEAL, WSP)	👍	varies	🤔	🤷	🤷

👍 good 🤷 partial 👎 poor 🤔 not measured ➖ not applicable

Who...	makes the data	sets the task and the judge	decides what is worth learning
Pretraining	humans	humans	humans
RL	model	humans	humans
Continual learning	model	model	model

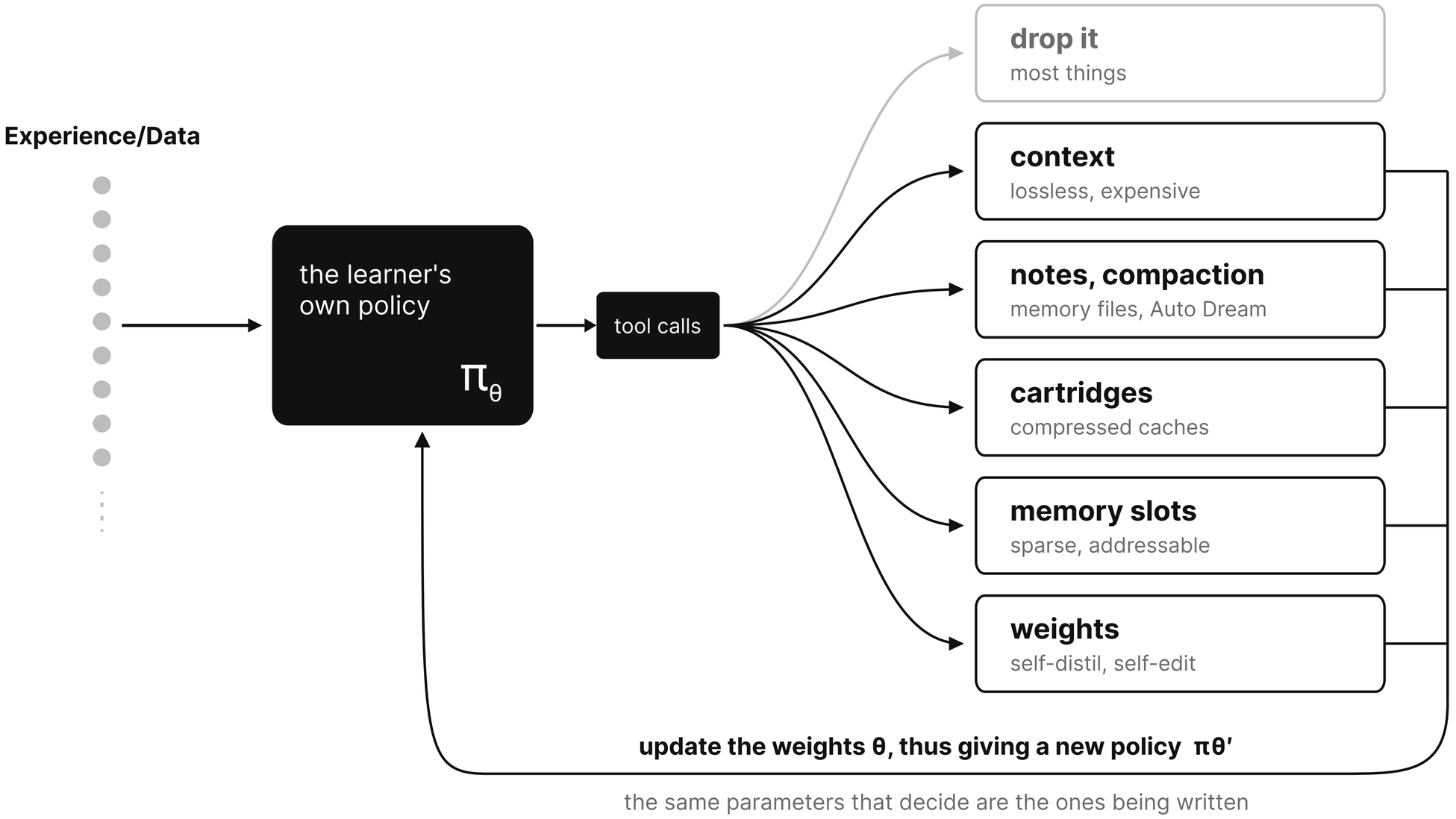
after Beren Millidge, “Three Phases of Capabilities and Alignment” (2026)

Out with all the talk that talks about what we do to the learner, in with all the talk that talks about what the learner does to itself



Slide credits: Hyung Won Chung

Out with all the talk that talks about what we do to the learner, in with all the talk that talks about what the learner does to itself



the same parameters that decide are the ones being written

What can close the loop? What work would be most useful?

That policy has to be trained somewhere, and very little today trains it over the horizons that matter.

Synthetic streams that resemble learning on the job, usable both as an RL environment and as an eval. Knobs like horizon, frequency, drift, facts vs skills.

Continual learning as rate–distortion *(h/t Karthik Balaji)*: under a budget of tokens, memory, weight updates and self-edits, keep whatever minimises error on future questions.

More benchmarks!

CL-Bench: Expert-checked tasks in six areas, including SWE, forecasting, game strats. Each area has tasks sharing a structure that an agent with persistent state can learn and a stateless one can’t.

Current findings: plain ICL beats dedicated memory. Sequences are anyway short and thus context can be lossless.

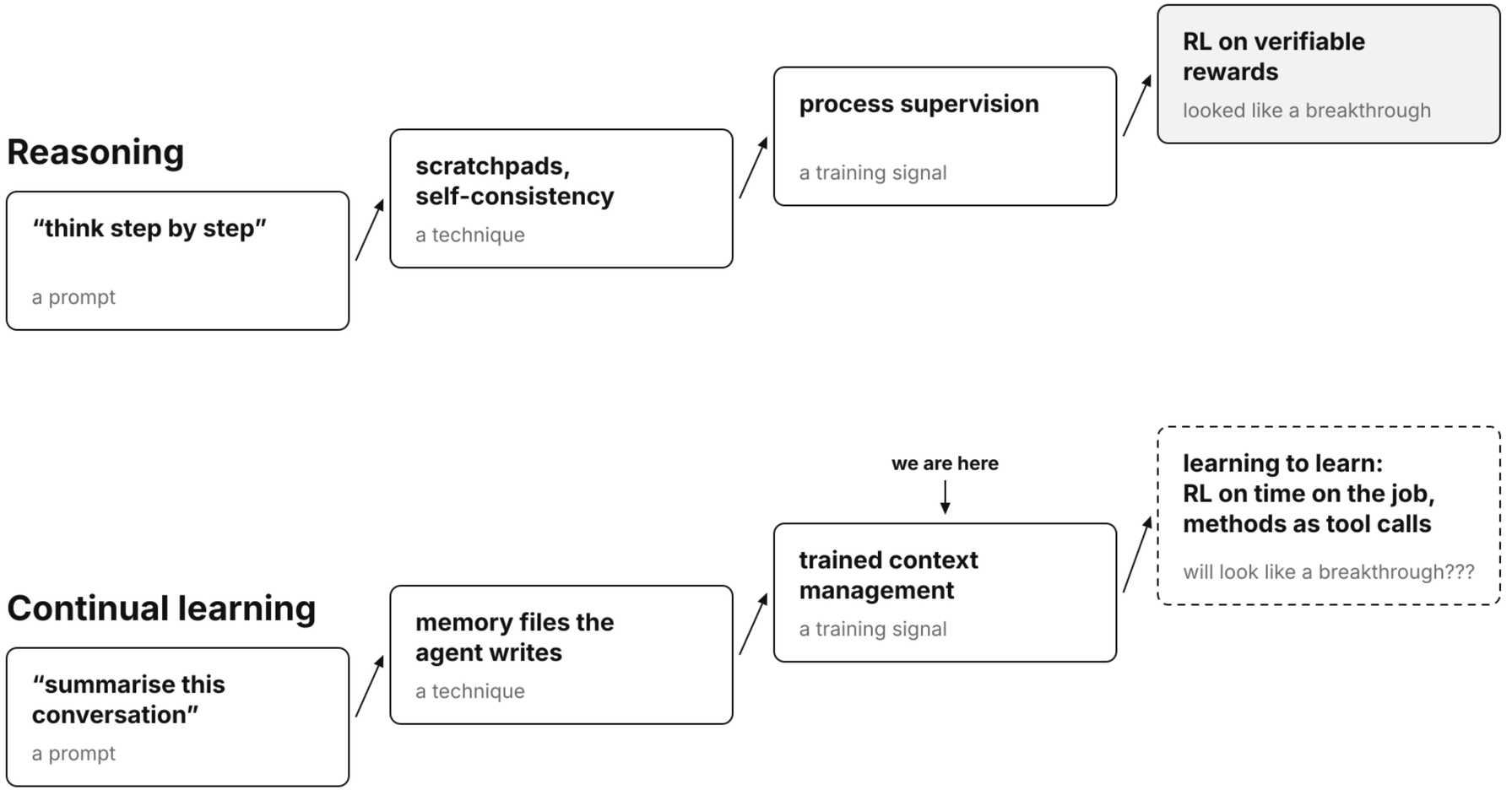
FutureSim: Replays 88 days of real news with O(100k) news articles. Agent has to forecast 330 questions that resolve in the period.

These are still short horizons!

More work on identifying tradeoffs between approaches quantitatively would be super valuable.

Asawa et al. “Continual Learning Bench: Evaluating Frontier AI Systems in Real-World Stateful Environments”
Goel, Chandak, Arun et al. “FutureSim: Replaying World Events to Evaluate Adaptive Agents”

very little exists, or will exist, the way it is thought to exist



Thank you.

My thoughts draw heavily on conversations with, and posts from, **Karthik Balaji** (@akbirthko),

and on posts from **Will Brown** (@willcb), **Charlie O'Neill** (@oneill_c), **Beren Millidge** (@BerenMillidge), **Jessy Lin** (@realJessyLin), **Ilija Lichkovski** (@carnot_cyclist) and **Andrew Lampinen** (@AndrewLampinen).

Slides with references at hiranmay.com/cl-talk

HMU: @hdarshane or darshanehiranmay@gmail.com