Information Constraints in Fine-tuning and Reasoning Language Models

Joey David

Master's thesis, M2 IASD, Université Paris Sciences et Lettres
Research internship at MILES, LAMSADE (CNRS, Université Paris Dauphine–PSL)
Supervised by Paul Caillon and Alexandre Allauzen

September 2026 · revised November 2026*

Abstract

This thesis studies how much information a language model must acquire or retain for a learned behavior to survive, across fine-tuning updates and intermediate reasoning states. For fine-tuning, we define an operational rate as the exact serialized size of a compressed LoRA or QLoRA adapter at a fixed level of retained held-out behavior. In these experiments, ordinary weight reconstruction error fails to predict behavioral preservation, and neither source description length nor the magnitude of the fine-tuning gain predicts the required rate. The ordering of the same corpora reverses across the frozen base models tested, indicating that adapter rate depends on the receiver as well as the data. Model-relative correction spectra provide the strongest predictor in the current campaign, although this remains a hypothesis rather than an established scaling law. On reasoning traces, different objectives select different partitions, and a state can be readily decodable without being reusable in downstream computation. Controlled explicit interfaces separate capacity from naming and update effects: enough bits are necessary, stable state names matter, and small one-step errors compound under repeated use. Together, these results suggest that useful information is not intrinsic to a dataset, adapter, or trace, but relative to a receiver, a task, and, under recursive use, an update rule.

Contents

1 Introduction **4**
1.1 Internship context . . . 4
1.2 Research trajectory . . . 4
1.3 Structure of the report . . . 4
1.4 Research questions . . . 5
1.5 Contributions . . . 5

2 Technical background **6**
2.1 Transformers and autoregressive language models . . . 6
2.2 Chain-of-thought and reasoning traces . . . 6
2.3 Fine-tuning, parameter efficiency, and LoRA . . . 7
2.4 Quantization and QLoRA . . . 7
2.5 Information, coding, and rate–distortion . . . 8

*This revision improves the wording and layout of the thesis submitted in September 2026. All results, figures and claims are unchanged. The fine-tuning work in Section 3 was later developed into the paper *The Behavioral Rate of Fine-Tuning: Compression as a Measure of Useful Information*, currently under review.

2.6 Internal representations, probes, and causal interventions . . . 8
2.7 Experimental conventions . . . 9

3 Fine-tuning under an information budget 9
3.1 Literature review . . . 9
3.1.1 Low-dimensional adaptation . . . 9
3.1.2 Quantization and quantization-aware adaptation . . . 9
3.1.3 Description length, bounds, and open measurement . . . 10
3.2 Initial pilots . . . 10
3.3 Fixed-checkpoint exact-rate protocol . . . 11
3.4 Adaptive allocation: a negative result . . . 11
3.5 Distinct examples at fixed compute . . . 12
3.6 Known-information control . . . 13
3.7 Searching for model-relative dataset information . . . 14
3.8 Takeaways . . . 16

4 Reasoning trajectories and objective-relative state 17
4.1 Literature review . . . 17
4.1.1 Written reasoning and action traces . . . 17
4.1.2 Hidden trajectories and mechanistic state . . . 17
4.1.3 Latent reasoning and reusable interfaces . . . 17
4.2 Trajectory pipeline and first hypotheses . . . 18
4.3 No universal partition of the token lattice . . . 18
4.4 Judged semantic spans . . . 19
4.5 Exact control: snapshot information is not memory . . . 19
4.6 Native language-model state and explicit handoff . . . 20
4.7 Rate, shared meaning, and closure . . . 21
4.8 Takeaways . . . 23

5 Synthesis and perspectives 23
5.1 A first bridge: reasoning and answer updates . . . 23
5.2 What the two tracks establish together . . . 24
5.3 What remains unresolved . . . 24
5.4 How to test the bridge . . . 24

6 Conclusion 25

A Additional experimental details and controls 26
A.1 Early adapter-factor asymmetry pilot . . . 26
A.2 Prequential-code implementation . . . 26
A.3 Codec ladder . . . 26
A.4 Exact diversity cells . . . 26
A.5 Adapter carrier and location controls . . . 26
A.6 Surface length and active update depth . . . 27

AI usage disclosure 29

List of Figures

1 Research sequence, April–August 2026. . . . 4
2 Low-rank adaptation of a frozen weight matrix. . . . 7

3 Adaptive MDL allocation and the RMSE counterexample. 12
4 Compute-matched distinct-example rate–retention curves. 12
5 Content diversity and distinct-example controls. 13
6 Known source information control. 14
7 Content diversity moves adapter rate on XBRL but not reliably on SQL. 15
8 Idealized solution object as a DAG. 18
9 Objective-relative partition utility and regret. 19
10 Exact closed-loop regret over horizon. 20
11 Native state factorization and explicit handoff. 21
12 Full-support proof-state rate control. 22
13 Local versus global register reliability. 23
14 Rate limits for reasoning and answer supervision. 24
15 Rank scaling and layer-band allocation controls. 27
16 Surface length, active depth, and behavior. 28

1 Introduction

1.1 Internship context

The internship took place from April to September 2026 at Université Paris Dauphine–PSL. I worked primarily from the PariSantéCampus offices, alongside other M2 students, PhD students, and researchers. As an M2 intern, I was part of *MILES (Machine Intelligence and Learning Systems)*, a research group within *LAMSADE*, the *Laboratoire d'Analyse et de Modélisation de Systèmes d'Aide à la Décision*, a joint research laboratory of Université Paris Dauphine–PSL and the CNRS.

LAMSADE is Dauphine's main computer science and decision-science research laboratory, spanning decision support, optimization, algorithms, data science, and artificial intelligence, while MILES focuses more specifically on the theoretical and algorithmic foundations of machine learning and artificial intelligence.

1.2 Research trajectory

The internship began with an information-theoretic study of fine-tuning, where early capacity and dataset-compressibility experiments exposed the need for an operational definition of update rate. The literature review also raised a parallel question about reasoning trajectories: do language models expose compact, reusable intermediate states? Those experiments increasingly showed that useful state descriptions depend on the downstream objective. They motivated more causal state-interface experiments and, in parallel, a return to fine-tuning with exact serialized-rate measurements. The two lines eventually converged on a common question: how much information must a representation preserve, relative to a particular receiver and utility, for learned behavior to survive? Section 5 presents a first experiment that applies the fine-tuning measurements to reasoning. Figure 1 summarizes this sequence of questions and experiments.

This question rests on a view that connects the two tracks: "useful information" is not an intrinsic property of a dataset, a fine-tuning adapter, or a reasoning trace. It is defined by a usage. Its amount depends on what the receiving model already knows, which behavior or objective must be preserved, and, for a state that will be reused, how that state is updated; from a classical compression point of view, it can be seen as a conditional description length: the information that remains necessary once the receiver and intended use are given as side information. The same artifact can therefore carry different amounts of useful information for different receivers or tasks. This view provides the common frame for the chronological experiments that follow: each experiment asks which information a specified receiver needs for a specified use, and which simpler account of that need the result rules out.

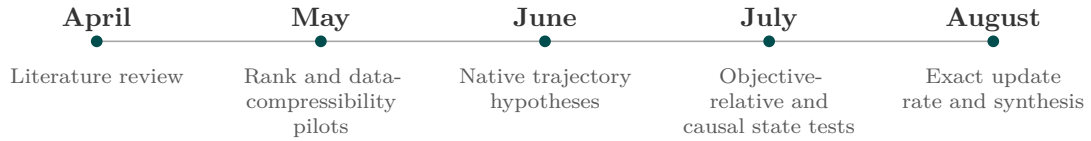


Figure 1: Approximate chronology of the research covered in this report. The internship ran from 13 April to 28 September 2026.

1.3 Structure of the report

The report aims to show the research process as well as its results, so negative results are kept: they often determined the next experiment, and sometimes changed the direction of the work. Section 2 introduces the technical background. Sections 3 and 4 cover the two research tracks,

fine-tuning under an information budget and reasoning trajectories; each opens with its own literature review and follows the order in which its experiments were run. Section 5 connects the two tracks, and Section 6 concludes.

1.4 Research questions

1. How should the rate of a fine-tuning update be defined when the receiver already has the base model, adapter layout, and decoder?

2. How much serialized update information preserves learned behavior, and which data or model properties predict that threshold, especially in the context of LoRA fine-tuning?

3. In the context of CoT-generating RLMs, do reasoning trajectories admit a canonical latent-space-based decomposition? If not, how variable are the decompositions across different utilities?

4. When can an explicit state code support RLM causal handoff and reliable recursive use?

1.5 Contributions

Our main contributions include:

1. **Weight-space fidelity does not determine preservation of learned behavior.** LoRA adapters with lower reconstruction error, e.g. via RMSE, can retain substantially less of the learned behavior than adapters with higher reconstruction error, showing that ordinary weight distortion is not an adequate measure of useful information. (Section 3.4)

2. **The accuracy gain of a model over a given dataset is a very poor predictor for the necessary amount of information of its fine-tuning update.** Across tasks, large behavioral improvements can correspond to very small behavior-preserving adapters: most notably, XBRL produces the largest fine-tuning gain in the campaign while requiring the lowest measured rate. (Section 3.7)

3. **The information written by fine-tuning is best expressed as relative to the frozen base model.** The same datasets require different adapter rates for different base models, and initial results suggest that model-relative properties of the corrections requested by the data predict this rate better than dataset size, base-model loss, or prediction gain alone. (Section 3.7)

4. **Chain-of-thought segmentation is strongly utility-dependent.** Boundaries that are useful for answer formation, symbolic updates, correctness, or compression differ substantially; no single segmentation performs well across all objectives. (Section 4.3)

5. **In RLMs, decodable intermediate states are not necessarily reusable causal states.** Native reasoning states could often be identified or decoded, but attempts to reuse them reliably in downstream computation were much weaker, suggesting that reasoning is not naturally organized as a sequence of clean, independently reusable states. (Section 4.6)

6. **Explicit state interfaces can support reliable multi-step reasoning, within limits.** When a compact state representation has enough capacity and one stable name per state, it supports substantially more reliable recursive computation than one-pass reasoning (97% against 72% final accuracy on a proof task). Small per-step errors still compound over long horizons. (Section 4.7)

2 Technical background

This section introduces the general concepts and notation needed for the rest of the report, and can be skipped by readers already familiar with the state of the art of reasoning language models, parameter-efficient fine-tuning, and representation analysis. The more specialized recent literature that directly motivates our experiments is deferred to the literature reviews in Sections 3 and 4.

2.1 Transformers and autoregressive language models

A language model assigns probabilities to sequences of discrete tokens. Modern large language models (LLMs) typically tokenize text into subword units, map each token to a vector representation, and process the resulting sequence through many transformer blocks (Vaswani et al. 2017). Each block combines self-attention, which lets a position aggregate information from earlier positions, with feed-forward transformations. Residual connections carry a vector-valued representation through the successive layers of the network.

For a prompt x and generated tokens $y_{1:T}$, an autoregressive model factorizes the probability of the continuation as

$$p_\theta(y_{1:T} \mid x) = \prod_{t=1}^{T} p_\theta(y_t \mid x, y_{<t}). \tag{1}$$

At each position, the final hidden representation is mapped to logits over the vocabulary and then to a probability distribution for the next token. Generation repeatedly samples or selects from this distribution and appends the chosen token to the context.

We write $h_t^{(\ell)} \in \mathbb{R}^d$ for the hidden representation at token position t and layer ℓ. For a fixed generated sequence, the collection $(h_1^{(\ell)}, \ldots, h_T^{(\ell)})$ can be viewed as a trajectory through representation space. This latent trajectory is distinct from the visible text sequence: two similar sentences can be represented differently internally, and a large internal change need not coincide with a linguistic boundary.

2.2 Chain-of-thought and reasoning traces

Chain-of-thought (CoT) prompting asks a language model to produce intermediate reasoning steps before its final answer. This simple change can substantially improve performance on multi-step reasoning tasks (Wei et al. 2022) and problem-solving more broadly. Models trained or prompted to produce such traces are often called reasoning language models (RLMs) in this report. Since 2022, however, chain-of-thought has evolved from primarily a prompting technique into a behavior deliberately induced during training. Methods such as reinforcement learning from verifiable rewards (RLVR) and optimization algorithms such as Group Relative Policy Optimization (GRPO) train models directly against the correctness of their final answers and can induce increasingly long, self-correcting reasoning traces without requiring explicit step-by-step demonstrations at inference time (DeepSeek-AI 2025; Shao et al. 2024). Modern reasoning models therefore commonly generate extended chains of thought by default, with the amount and structure of intermediate computation becoming part of the learned policy rather than merely a consequence of prompt engineering. Post-training can also give CoT a distributional form quite unlike ordinary language generation: the trace is optimized primarily as an internal computational workspace rather than as communicative prose, which can make stylistic or formatting interventions brittle. Although it is produced by the same autoregressive mechanism, this different functional role gives reason to expect correspondingly different latent dynamics.

A CoT trace gives two related objects to study. The first is the written trajectory: the sequence of generated tokens, sentences, equations, or explicit intermediate claims. The

second is the latent trajectory formed by the model's hidden representations while generating those tokens. The written trace is directly observable, while the latent trajectory requires access to model activations. Nothing in the autoregressive architecture requires meaningful computational steps to coincide with sentences, punctuation, or any other fixed textual unit.

2.3 Fine-tuning, parameter efficiency, and LoRA

Pretraining produces a general-purpose parameter vector θ_0. Fine-tuning adapts that model to a new dataset or task by continuing optimization on a new objective. Where full fine-tuning updates most or all model parameters, parameter-efficient fine-tuning instead keeps most of the pretrained model fixed and learns a much smaller task-specific object.

Low-Rank Adaptation (LoRA) is a widely used example (Hu et al. 2022). For a pretrained weight matrix $W \in \mathbb{R}^{d_{\text{out}} \times d_{\text{in}}}$, LoRA freezes W and learns a low-rank update

$$W' = W + \Delta W, \qquad \Delta W = \frac{\alpha}{r} BA, \tag{2}$$

where $A \in \mathbb{R}^{r \times d_{\text{in}}}$, $B \in \mathbb{R}^{d_{\text{out}} \times r}$, and r is much smaller than the dimensions of W. LoRA modules can be inserted into several projections of each transformer block while the underlying pretrained weights remain unchanged.

This gives a useful separation between a *base model*, which is shared, and a comparatively small learned *adapter*, as illustrated in Figure 2. Rank controls the dimension of the update parameterization, but it should not be confused with a number of information bits: the learned matrix entries still need numerical values, and many different numerical representations are possible at the same rank.

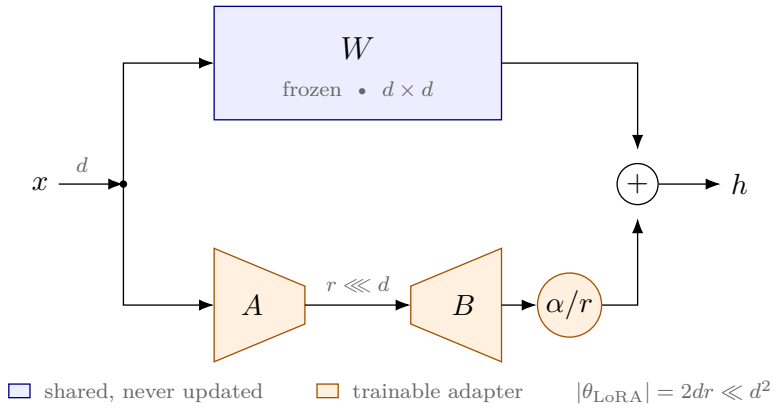


Figure 2: LoRA freezes the pretrained weight matrix W and learns the low-rank adapter update $\Delta W = (\alpha/r)BA$.

2.4 Quantization and QLoRA

Neural-network parameters are usually stored as floating-point numbers. Quantization represents them with a smaller finite set of values. A simple quantizer stores, for example, an integer code for each weight together with one or more scale parameters that map the codes back to approximate real values. Fewer available levels reduce memory use but increase numerical distortion.

Quantization can be applied to different objects for different reasons. A pretrained backbone may be quantized to reduce memory during inference or fine-tuning, while a separately learned

adapter may remain in higher precision. Representative methods include GPTQ (Frantar et al. 2023), ZeroQuant (Yao et al. 2022), SmoothQuant (Xiao et al. 2023), and AWQ (Lin et al. 2024) for post-training or deployment quantization. QLoRA uses the same separation during adapter training: it keeps a pretrained model in a four-bit quantized representation while training LoRA adapters through it (Dettmers et al. 2023). Its NormalFloat-4 (NF4) representation is designed for weight distributions encountered in pretrained networks.

2.5 Information, coding, and rate–distortion

Information theory provides a language for relating probability, coding, and compression. If an event x has probability $p(x)$, its surprisal in bits is

$$-\log_2 p(x). \tag{3}$$

The entropy of a random variable is the expected surprisal. In an ideal code, more probable events can be represented with shorter descriptions, so negative log-probability can also be interpreted as an idealized code length. For a language model, average negative log-likelihood therefore measures both predictive error and the number of bits per token that the model would need in an ideal probabilistic code.

Conditional quantities express the value of information when some side information is already known. If both sender and receiver already possess a pretrained model, for example, our interest would lie in the number of additional bits needed to specify whatever changes relative to it, rather than how many bits are required to describe that model again. *Mutual information* similarly measures how much knowing one random variable reduces uncertainty about another.

Rate–distortion theory asks how small a representation can be while keeping some notion of error below an acceptable level (Cover and Thomas 2006). The key point is that a rate has meaning only together with a distortion measure. Two compressed objects with the same numerical reconstruction error can behave differently if errors occur in different functionally important directions. Conversely, a representation may differ substantially in Euclidean space while preserving the behavior that matters.

Minimum Description Length (MDL) uses description length as a way of expressing regularity: a model or explanation is useful when it gives a short joint description of the model and the observations (Grünwald and Roos 2019). In this report, MDL and rate–distortion ideas provide conceptual motivation, while the exact rate and distortion that ended up being most widely used for the LoRA experiments are defined in Section 3.

2.6 Internal representations, probes, and causal interventions

As introduced above, a *hidden representation* is the vector state of the residual stream after one of the transformer's attention and feed-forward blocks. Hidden representations can contain information that is not explicit in the generated text. A common way to test this is a *probe*: a simple supervised model, often linear, is trained to predict a property of interest from hidden activations. High held-out probe performance shows that the property is statistically accessible from the representation under the chosen decoder.

Geometric analyses ask related but unsupervised questions. Principal Component Analysis (PCA) can identify high-variance directions; cosine similarity measures changes in direction; distances or local statistics can reveal how a trajectory evolves through representation space. Such measurements describe organization in the representation but do not by themselves show that the model uses that organization causally.

Where probes offer evidence for decodability, causal interventions provide stronger evidence. In activation patching or interchange experiments, an internal activation from one computation

is replaced by an activation from another and the downstream output is observed. If changing a representation changes later behavior in the way predicted by its proposed semantics, this supports a causal role. It is therefore useful to keep three claims separate: a variable may be *decodable*, may form a clear *geometric structure*, and may be *causally used*; importantly, none of these necessarily implies the others.

2.7 Experimental conventions

For language modeling, we frequently report held-out negative log-likelihood (NLL), often converted to bits per token using base-two logarithms. This is a dense metric as every scored token contributes an observation. We also resort to exact or regex-based string matching for tasks with a small number of possible outputs, such as multiple-choice questions or arithmetic problems. In these cases, we report the fraction of correct answers.

Controlled comparisons also require matching the resource that is meant to stay fixed. Depending on the experiment, this can mean the same number of optimizer updates, examples, tokens, parameters, or representational capacity. We state the matching rule with each experiment rather than assuming it from a shared nominal setting.

3 Fine-tuning under an information budget

3.1 Literature review

3.1.1 Low-dimensional adaptation

Intrinsic-dimension results show that successful language-model adaptation can lie in a much smaller subspace than the full parameter space (Aghajanyan et al. 2021). The original LoRA method leverages that constraint, by learning a low-rank update to selected weight matrices (Hu et al. 2022). On top of LoRA, AdaLoRA allocates rank by an importance score rather than fixing the same rank throughout the model (Zhang et al. 2023). Work on LoRA factor asymmetry further shows that the input and output factors can contribute unequally despite having the same rank and similar storage cost (Zhu et al. 2024).

3.1.2 Quantization and quantization-aware adaptation

Whole-model quantization provides several ways to allocate error. GPTQ uses approximate second-order information during sequential weight quantization (Frantar et al. 2023). ZeroQuant uses hardware-aware groups and layer-wise distillation (Yao et al. 2022); SmoothQuant moves activation difficulty into weights before applying integer quantization (Xiao et al. 2023); and AWQ preserves channels selected from activation statistics (Lin et al. 2024). These methods show that equal nominal width does not imply equal functional cost across weights or layers.

QLoRA trains higher-precision adapters over a four-bit frozen backbone (Dettmers et al. 2023). LoftQ chooses an initialization that reduces joint quantization and low-rank approximation error (Li et al. 2024), while QA-LoRA aligns quantization groups with the adapter so that the learned change can merge into a low-bit model (Xu et al. 2024). Their main targets are training memory, initialization error, or low-bit deployment. They do not directly measure the shortest complete file that reproduces the behavior of one fixed trained adapter.

3.1.3 Description length, bounds, and open measurement

Rate–distortion theory formalizes the least message rate compatible with a chosen distortion (Berger 1971; Cover and Thomas 2006). Minimum Description Length (MDL) treats short joint descriptions as a model-selection principle (Grünwald and Roos 2019). Several more recent results make these ideas concrete in neural networks.

Young (2025) takes the rate–distortion view literally in *Radio*, framing post-training LLM quantization as a trade-off between the number of transmitted bits and the distortion induced by compression. Their method can target either a desired compressed model size or a tolerated loss of accuracy. Their framing of compression as an allocation problem under a global rate budget highlights that a bit need not have the same functional value everywhere in a network.

Liu et al. (2025) give a complementary empirical picture with *ParetoQ*, a unified framework for comparing 1-, 1.58-, 2-, 3-, and 4-bit quantization. They report a marked transition between two and three bits: at three bits and above, fine-tuned quantized models remain comparatively close to the pretrained distribution, whereas at two bits and below the learned representations change much more substantially. Their size–accuracy results also show that lower nominal precision does not translate monotonically into worse trade-offs. This hints at bits per parameter not being a linear measure of usable model capacity; and at the behavioral meaning of a rate depending on representation and training regime.

A different connection between compression and learning appears in the older work of Arora et al. (2018). They construct explicit succinct reparameterizations of trained deep networks and use them to obtain generalization bounds substantially stronger than naive parameter counting. They rely on noise-stability properties, that let the compressed network approximately preserve the original predictions. Later PAC-Bayes work develops related compression-based bounds (Zhou et al. 2019; Lotfi et al. 2022).

Finally, Bergsträßer et al. (2026) study description length from the opposite direction in *Transformers are Inherently Succinct*. In a formal-language setting, they prove that fixed-precision transformers can represent some languages exponentially more succinctly than linear temporal logic or recurrent networks, and doubly exponentially more succinctly than finite automata. Though not about fine-tuning, this theorem gives a useful intuition for our setting: the frozen transformer is a powerful decoder rather than passive side information, and a small update may select or specify a behavior whose standalone description would be far larger. This makes the relevant quantity a description *relative to the base model* rather than an intrinsic description of the task.

3.2 Initial pilots

Throughout this section, a training set is a collection of supervised prompt–response *examples*. We call one model–dataset–training condition an experimental *arm*, and a related collection of arms a *panel*.

Once our main literature review phase was over, our first hypothesis was that LoRA rank could serve as a cheap measurable proxy for how much a fine-tune writes: if useful adaptation really occupies a small subspace, the smallest effective rank might behave like a crude information budget. To test it, we began with a small QLoRA rank sweep before building a true codec.

An initial Qwen2.5-0.5B sweep used instruction-following examples from the *Alpaca* and *Dolly* datasets, with LoRA ranks $\in \{0, 2, 4, 8, 16\}$; rank 0 being the frozen-model baseline. Most improvement in held-out next-token loss appeared between rank 0 and rank 2, with smaller gains thereafter. Useful adaptation occupied a small low-rank subspace in these runs.

On top of rank, we also wondered whether the amount or ordinary compressibility of the training data predicted how much information would be written. Our pilot dataset

variants either duplicated examples or made their prompt–response templates more regular. Exact duplication is deliberately content-free, as it increases the nominal number of stored examples without adding a new training case. But under epoch-based training, it also changes optimization exposure and revisiting frequency.

Although these pilots were imprecise and mostly negative, they ruled out several easy explanations: nominal rank is not a rate, nominal example count is confounded with training exposure, superficial corpus compressibility is not a clean information intervention, and a small exact-accuracy benchmark is too noisy. This led to the next question: what happens if we hold one learned update fixed and vary only how many bits are used to store it?

3.3 Fixed-checkpoint exact-rate protocol

To measure an actual rate–distortion curve, we train one rank-16 LoRA adapter and then keep that checkpoint fixed. Every point on the curve is therefore a different compressed representation of the same learned update, rather than a separately trained model.

Our codec quantizes the LoRA factors, bit-packs the quantized values together with their scales and metadata, compresses the payload with zlib, and writes the result to disk. We then reload that file into the frozen base model before evaluation. The reported rate is the complete serialized file size divided by the number of values in the original adapter, in bits per adapter value.

The exact codec ladder is described in Section A.3.

We measure behavior mainly with held-out next-token negative log-likelihood. If L_0 is the frozen-model loss, L_{raw} the loss of the uncompressed adapter, and L_r the loss after compression to rate r, retained gain is

$$\text{Retain}(r) = \frac{L_0 - L_r}{L_0 - L_{\text{raw}}}.$$

We use $R^\star(.90)$ for the smallest measured rate that retains at least 90% of the uncompressed adapter's gain. The 90% threshold is simply a consistent operating point for comparing experiments; the full rate–retention curves remain available when the threshold hides useful detail.

3.4 Adaptive allocation: a negative result

Once the rate axis was defined, we asked whether a fixed bit budget could be allocated more intelligently than uniform precision. Weight reconstruction error was a deliberately simple first proxy: if the rows that are hardest to reconstruct are also the rows whose corruption matters most for behavior, an error-aware allocator should win at the same file size.

To test this hypothesis, the adaptive codec could drop each row of a LoRA factor matrix or encode it at 1, 2, 3, 4, or 8 bits. A Lagrangian objective traded reconstruction error against code length and chose the allocation. Contrary to the hypothesis, this did not beat uniform coding: as Figure 3 shows, an adaptive point had lower adapter RMSE than a fixed-rate point but preserved far less of the learned behavior.

Scientific conclusion. This result rules out global weight error as a sufficient measure of useful adapter information. Two reconstructions can have similar size or weight error yet differ sharply in the behavior they preserve for the frozen model, so the relevant distortion must be measured at the receiver's output. This shifted the question from *where* to spend a fixed budget to what determines how large the required budget is in the first place.

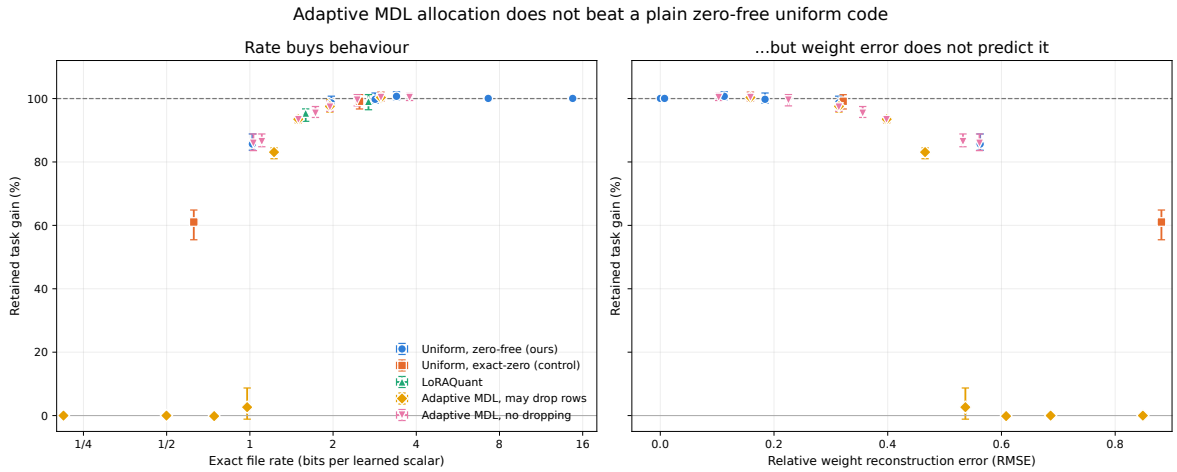


Figure 3: Adaptive MDL allocation against fixed-rate controls. Left: retained held-out prediction gain against exact decoded file rate. Right: retained gain against relative weight RMSE.

3.5 Distinct examples at fixed compute

Our next hypothesis was that the required rate grows with the amount of distinct training data. To test it without changing training exposure, we varied the number of distinct MetaMathQA examples from 2,000 to 32,000 while holding both optimizer steps (2,000) and total example presentations (32,000) fixed.

As Figure 4 shows, $R^\star$ generally increases with the number of distinct examples: the fit $R^\star = -0.26 + 0.081 \log_2 n_{\text{distinct}}$ gives $R^2 = 0.78$ over 15 runs. Though the trend is not perfectly monotone, it can be used as a rough scaling law. In the most repeated conditions, compression can even outperform the raw adapter on held-out loss, suggesting that it removes some overfit components. This observation supports a broad relationship between distinct training data and required adapter rate; exact cells appear in Table 3.

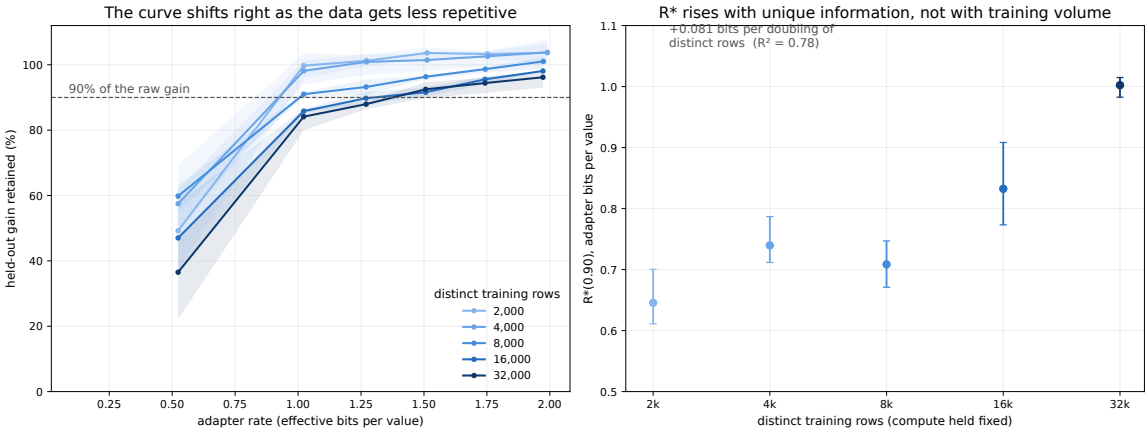


Figure 4: Compute-matched distinct-example experiment. The left panel shows the full rate–retention curves; the right panel shows the derived $R^\star(.90)$ cells and the fit over 15 runs.

Distinct examples, however, are only a rough proxy for distinct content. Two examples can be different records while still expressing the same underlying problem. We therefore tested the narrower hypothesis that content diversity determines the rate, using exactly 8,000 training examples in every condition but varying how many source math problems those examples came from. As Figure 5 shows, increasing the number of source problems from roughly 400 to 5,620 raised mean $R^\star$ from 0.682 to 0.742 bits per value across six seeds.

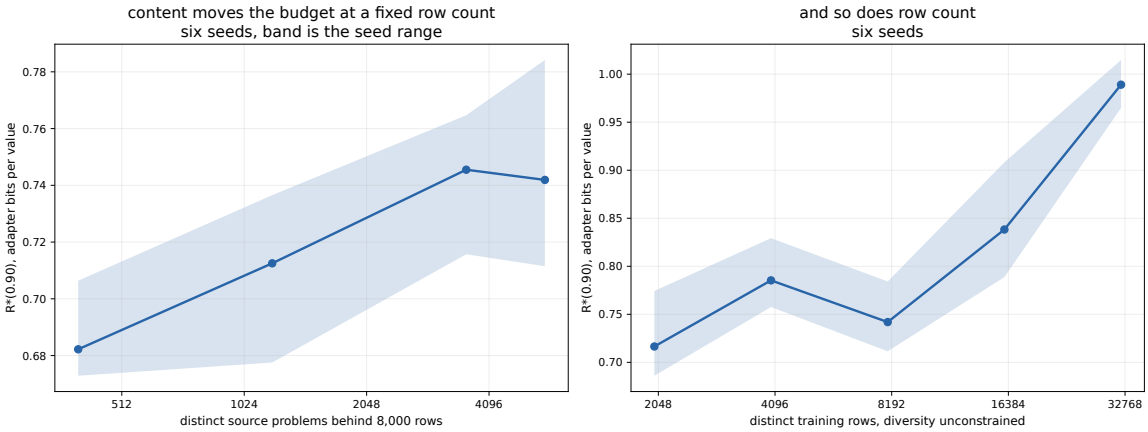


Figure 5: Two related but distinct diversity controls. Left: content diversity changes behind a fixed 8,000-example corpus, with six seeds. Right: the number of distinct training examples changes while optimization exposure is fixed, with six seeds in the updated reduction.

The right-hand control in Figure 5 holds optimization exposure fixed, ruling out extra training steps as the cause of the distinct-example trend. The left-hand control holds example count fixed and therefore isolates source-problem diversity from nominal dataset size. It does not, however, rule out another explanation: the adapters trained on more diverse data also tended to improve the model more. Retaining 90% of a larger improvement may itself require a larger file. We therefore asked how to separate source information from the amount of behavior learned.

3.6 Known-information control

Since natural corpora do not come with a known number of information bits, a failed correlation could always be blamed on a poor information proxy. To remove that ambiguity, we built a synthetic mapping task whose source information is known by construction. Each prompt maps to one of 16 one-token labels, and the conditions differ in the number of independent four-bit label assignments used to generate the data.

We also estimated the dataset's description length with a conditional *prequential* code, which trains the model on earlier blocks of data and uses its loss to encode later ones (Section A.2). A plain prequential code over-counts when the model is confidently wrong, so we added a uniform fallback and a switch-code correction that lets a block fall back to the frozen base model when that is cheaper.

After that correction, the prequential code orders the 4-, 64-, 128-, 256-, and 512-bit conditions correctly and stays within 1.1–3.5 times the known source count, as shown in Figure 6. The observation rules out a wholly uncalibrated source-information proxy, although it does not yet explain adapter rate.

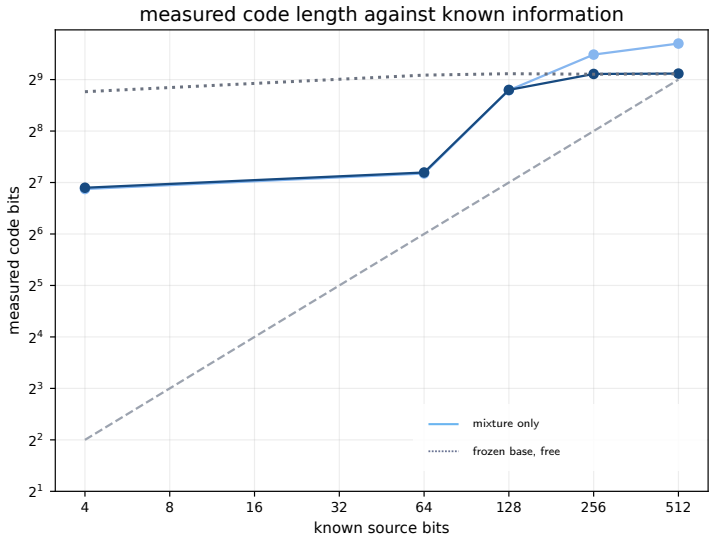


Figure 6: Known-information control. The conditional code recovers the correct order of source information.

3.7 Searching for model-relative dataset information

In line with the original internship proposal, the synthetic result suggested that the relevant information might be conditional on what the frozen model already "knows". Before introducing a new model-relative measure, however, we eliminated two simpler explanations: perhaps the rate is just determined by how much the fine-tune improves the model, or perhaps each dataset simply has an intrinsic difficulty.

The first math experiments seemed to support the first explanation: across 48 rank-16 runs, held-out prediction gain explained 56% of the variation in $R^\star$. But that relationship covered only a narrow range of gains, so it did not tell us whether gain itself determined the rate.

We first rewrote the same target answers in five deterministic styles, ranging from short plain responses to symbolic ones. This design changes how much next-token behavior the adapter learns while leaving the underlying answers fixed. The rewrites nearly doubled the learned prediction gain, from 0.555 to 1.062 bits per token, while $R^\star$ moved only from 0.684 to 0.763.

We then ran a separate 16-fold optimizer-budget sweep. This changed the amount learned while holding the data content fixed, and again failed to produce the expected rate increase.

The two controls vary learned gain in different ways. Together they rule out behavioral improvement alone as the determinant of required rate, though they do not identify what replaces it. We therefore next tested the second simple explanation: that each dataset has an intrinsic adapter cost.

We next changed the frozen model while keeping the fine-tuning setup fixed. The same five corpora were trained on Mistral-7B and Qwen2.5-7B with the same LoRA rank, number of examples, epochs, and codec. If a corpus had an intrinsic adapter cost, its ordering should remain similar across the two receivers. It did not: only dialogue kept the same position, while instruction tuning, summarization, and math changed substantially in relative cost.

Scientific conclusion. Changing the frozen model is a direct change of receiver. The reversal in corpus ordering therefore establishes that a dataset does not impose one intrinsic adapter cost: the cost depends on which model must use the update.

The earlier math experiments still suggested that content diversity can matter, so we tested

Corpus	Mistral-7B $R^\star(.90)$	Qwen2.5-7B $R^\star(.90)$
Instruction	0.643	0.774
Summarization	0.652	0.388
Math	0.737	0.411
Code	0.891	0.668
Dialogue	0.913	1.176

Table 1: The corpus ordering changes between frozen base models. Each cell averages three independently trained adapters; all 30 rate curves bracket the target.

that effect on two structured tasks, text-to-SQL and XBRL tag extraction, with both base models. We held the total number of training examples fixed while varying how many SQL domains or XBRL companies were represented in training. XBRL shows the same direction in all six matched model–seed comparisons; SQL changes much less and inconsistently across seeds.

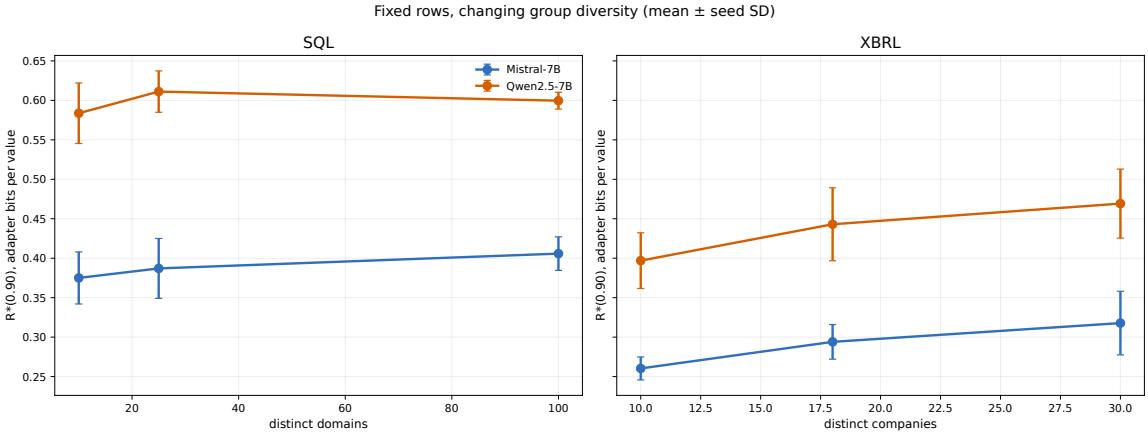


Figure 7: Fixed-example diversity panel. Points show means and bars show one standard deviation over three seeds.

The SQL and XBRL observations are compared in Figure 7, with exact cells in Table 4. The XBRL intervention is not a clean measure of abstract diversity: increasing the number of companies also changes which companies the adapter sees during training, while the test set covers all companies. The result may therefore partly reflect support coverage. Nearly constant answer entropy rules out a simple label-entropy explanation, but a cleaner diversity intervention is still needed and was left for future work.

By this point, the most plausible remaining quantity was the set of corrections that the dataset asks a particular frozen model to make, so we looked for a measure of it. Our most useful family of measures came from the token gradients. For a scored token, the output-head gradient factorizes as

$$g_{it} = h_{it} \otimes (p_{it} - e_{y_{it}}),$$

where h_{it} is the hidden state, p_{it} the model's predicted distribution, and $e_{y_{it}}$ the correct target. Intuitively, one factor describes the representation the model is using and the other describes how its prediction needs to change.

We project these factors to a manageable dimension and summarize the spectrum of their empirical second moment $F = n^{-1}G^\top G$. Our main scalar is

$$I_{\text{corr}}(D; M) = \log\det(I + F) = \sum_j \log(1 + \lambda_j(F)).$$

This quantity grows when the requested corrections are large and spread over many independent directions. Crucially, it depends on both the dataset D and the frozen model M: changing the receiver changes its hidden states, its errors, and therefore the correction spectrum.

After averaging seeds and removing exact data reuses, our comparison contains 22 distinct model–dataset conditions across seven task families. We evaluate each candidate both by how well it orders $R^\star$ within a model and by how well it predicts a task family left out of fitting. Because ten measures were screened on the same data, our reported permutation test also corrected for selecting the best candidate from that set.

Measure	Within-model ρ	Adjusted p	Held-out RMSE	Pooled ρ
Correction Fisher log-volume	**0.764**	**0.00034**	0.187	0.584
Correction Fisher effective rank	0.609	0.0112	**0.181**	**0.814**
Surprise-weighted hidden log-volume	0.614	0.0103	0.204	0.441
Base-model target code length	0.200	0.524	0.261	0.269
Model-only fit	—	—	0.264	—

Table 2: Screen of ten model-relative measures against likelihood-based $R^\star(.90)$. RMSE leaves one task family out. ρ is the Spearman correlation, and p is the p-value of the permutation test.

The correction spectrum also follows the earlier SQL/XBRL contrast: its log-volume is nearly flat as SQL diversity increases but rises with XBRL diversity on both base models. It is not simply another name for learning gain; across the distinct conditions, correction log-volume and held-out gain have Spearman correlation -0.275.

This is our strongest predictor. However, it uses a projected output-head gradient rather than the full LoRA Fisher, and the study contains only two 7B receivers. We therefore treat the correction spectrum as a promising model-relative hypothesis, not yet as an established scaling law.

Additional controls vary adapter rank and the depth band from which bits are removed. They support the same conclusion—the measured rate depends on its carrier, and equal weight error does not imply equal behavioral cost—but they are secondary to the model-relative result and appear in Section A.5.

3.8 Takeaways

Our clearest methodological result is straightforward: if we want to ask how much information a fine-tune writes, we should measure the size of an actual reloadable update that preserves a stated behavior, rather than infer it from LoRA rank or weight error.

The experiments then rule out several simple predictors of that rate. It is not set by source entropy, base-model loss, or the magnitude of the behavioral improvement alone. The same corpus can require different rates on different frozen models, and content diversity matters in some tasks but not others.

The results also point to a model-relative explanation: the rate tracks the number and magnitude of correction directions that the training data asks the frozen model to make. That result is promising but still needs prospective replication.

4 Reasoning trajectories and objective-relative state

4.1 Literature review

4.1.1 Written reasoning and action traces

Chain-of-thought prompting elicits written intermediate steps before an answer and improves several reasoning tasks at sufficient model scale (Wei et al. 2022). ReAct extends the trace with actions and observations, so intermediate text supports both deliberation and interaction with an environment (Yao et al. 2023). These methods establish the practical value of extra computation in the sequence. They do not show that each written step matches a discrete internal update or that the text faithfully records the cause of the answer.

Work on efficient entity tracking makes the state problem more exact. In the formal setting studied by Fagnou et al. (2024), whose authors include my supervisors, standard attention needs growing depth to follow a chain of state changes, while their modified attention composes the chain more directly.

4.1.2 Hidden trajectories and mechanistic state

How CoT improves problem solving is still an open research question. Composition of abstract claims is a plausible part of the answer, but it has not been clearly shown. Mechanistic analyses ask which internal components move or write task information. On fictional ontology tasks, Dutta et al. (2024) report a split between earlier relation transfer and later answer-writing paths. Across abstract rule and analogy tasks, Yang et al. (2025) identify early symbol abstraction, intermediate induction, and late retrieval components. These studies support structured, task-linked computation, but their units come from the chosen task and intervention target.

Trajectory work shifts the unit of analysis from a component to a position or step. Qian et al. (2025) identify "thinking tokens" with high estimated mutual information, Yu et al. (2025) study state-aware changes across reasoning stages, and Sun et al. (2026) report step-specific geometry and correctness signals. The methods differ in target, estimator, and granularity. Their findings motivate a direct test of whether boundaries selected for answer formation, symbolic updates, correctness, and geometric compression agree when the token budget stays fixed.

4.1.3 Latent reasoning and reusable interfaces

Continuous latent reasoning removes the requirement that every intermediate state decode to text. Coconut feeds hidden states back into the model as continuous thoughts and trains their repeated use (Hao et al. 2025). A later preprint uses causal perturbations and adversarial tests to argue that apparent latent reasoning can instead rely on weakly meaningful placeholders in some settings (Zhang et al. 2025). This disagreement makes output quality alone insufficient evidence for a reusable state interface.

Prior work thus finds useful tokens, stages, circuits, and latent states, but each selects an often new notion of state through its task and measure. Studying what the information-theoretic limits of fine-tuning imply for RLMs raised two questions: whether one token decomposition remains useful across several objectives, and whether a decoded or produced state remains sufficient after the source history is removed and the state is used again. We began with the simpler idea that a long verbal derivation might hide a short series of stable updates.

4.2 Trajectory pipeline and first hypotheses

Our initial hypothesis was that a long chain of thought might follow a *solution object*, composed of compact internal updates corresponding to meaningful steps of the reasoning. If such objects existed naturally, several things should happen at roughly the same places in the trace. A meaningful text boundary should coincide with a localized latent change; equivalent computations should permit causal transfer; similar operations should have similar representations; and those units should help predict whether the solution is correct.

To test this hypothesis, we looked for global, defensible, utility-neutral partitions of CoT traces, following the idealized structure in Figure 8. Such partitions would have allowed us to split CoT streams into chunks that could then be classified as different types of reasoning operations.

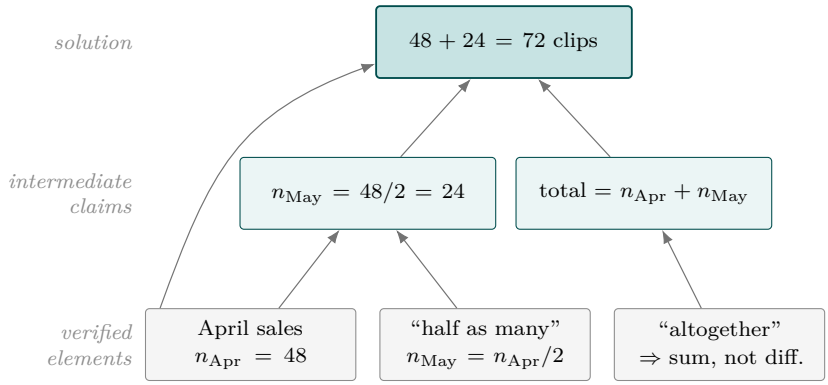


Figure 8: *Problem. Natalia sold clips to 48 of her friends in April, and then she sold half as many clips in May. How many clips did Natalia sell altogether in April and May?*
A hypothetical solution object viewed as a directed acyclic graph over claims. Leaves are directly verifiable readings of the problem statement; internal nodes are operations that consume earlier claims; the sink is the answer.

4.3 No universal partition of the token lattice

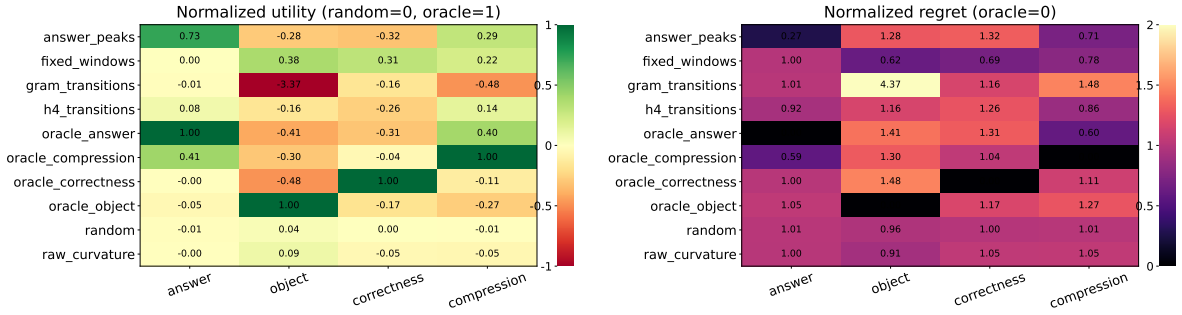
For this comparison, every gap between two generated tokens could be chosen as a boundary. Each trace received the same number of boundaries, based on its number of verified symbolic updates, so one method could not win simply by selecting more points. For each objective, an oracle was allowed to see the true objective scores and choose the best boundaries; random selection was normalized to zero and the oracle to one.

We used four objectives. The first rewards boundaries where the representation moves toward the gold answer; the second rewards completion of a verified symbolic update; the third rewards changes in a held-out correctness probe; and the fourth rewards changes in local PCA geometry, used as a compression-oriented signal. We then asked whether the boundaries that are good for one objective are also good for the others.

The observation, summarized in Figure 9, is that they mostly are not. On 310,044 held-out SmolLM3-3B tokens and 61,691 Qwen3-14B tokens, overlap between objective-specific oracle boundaries is low: F1 within four tokens ranges from 0.098 to 0.210 for SmolLM and from 0.130 to 0.294 for Qwen. No single rule performs well on all four objectives: the best worst-objective utility is only 0.020 for SmolLM and 0.080 for Qwen. Sentence boundaries are particularly weak for answer and symbolic-update utility, although raw cosine peaks retain some answer-related signal.

The negative result is not caused by an absence of learnable structure. Supervised detectors recover their own objective's boundaries with held-out ROC–AUC of 0.999/0.984/0.849/0.868

for SmolLM and 0.998/0.992/0.859/0.825 for Qwen (answer, symbolic update, correctness, and compression). In other words, each target leaves a strong enough signal to decode, but the signals select different partitions of the same trace.



(a) Normalized utility; random is zero and the target oracle is one.

(b) Normalized regret; the target oracle is zero.

Figure 9: Objective-relative partitions on the primary SmolLM corpus.

Scientific conclusion. The high detector AUCs establish that reasoning traces contain several readily decodable structures; the low cross-objective transfer establishes that these structures select different partitions. The result is therefore not an absence of latent structure, but an absence of one utility-independent structure. This shifted the project from searching for one canonical partition to asking what kind of state is sufficient for a specified future use.

4.4 Judged semantic spans

However, our four objectives were deliberately operational and might simply have missed the semantic steps that a human reader would naturally call reasoning. To test this, a larger model acting as a judge, Qwen3.5-122B-A10B, labeled 415 overlapping windows from concise correct SmolLM traces with nine edit types: *orient*, *bind variable*, *add relation*, *derive value*, *verify*, *correct error*, *plan*, *extract answer*, and *mixed*. After alignment and edge filtering, the audit retained 1,761 labeled spans over 58 traces, with median length 29 tokens. These semantic boundaries are strongly visible in the hidden states. A question-disjoint latent detector reaches AUC 0.982 overall and 0.969 away from sentence endings. The question-disjoint split rules out memorizing the same problems, and the away-from-ending score rules out sentence punctuation as the whole signal. However, lexical TF–IDF alone reaches 0.892, so the experiment does not establish that the decoded structure is uniquely latent rather than also expressed in the words themselves.

Most importantly, these human-like semantic boundaries still transfer poorly to the four earlier objectives: their utilities are $-0.012/0.043/0.036/0.324$. A readable semantic decomposition therefore exists and is useful, but it is not a unique decomposition privileged by answer formation, symbolic updates, correctness, and compression.

4.5 Exact control: snapshot information is not memory

The partition experiments led to a new hypothesis: a representation that contains the right information *now* may not remain useful after the system continues operating. A state can be perfectly decodable at one instant and still be a bad memory if its update rule forgets important information later.

To isolate that distinction from language-model noise, we built an exact finite control, in the same fashion as our synthetic fine-tuning task from Section 3. In this task, a controller

observes a short history generated from a finite hidden state and must make decisions over a future horizon using one of several memory representations. Because the hidden-state model, memory update, and planner are all explicit, any difference in performance can be attributed to the representation and its update rule rather than to generation or judging errors.

The observation in Figure 10 is that, across 90 task cells, the full public history and a 12-bit text suffix start with exactly the same support, state mutual information (2.512 bits), and path leakage (1.781 bits), yet never match in regret: pooled regret with the text suffix is 7.19 times higher. At horizon 32, regret is 2.20 with full history and 30.91 with the suffix. Both representations contained the same initial observations, but the suffix discarded older information as new observations arrive.

A 12-bit explicit field state performed much better: it removed 79.3% of the regret of having no memory, compared with 93.7% for full history and 54.8% for the text suffix. Matching the initial support, mutual information, and path leakage rules out initial capacity as the explanation for the suffix's higher regret; the difference appears only as the memories are updated. The field state still failed when the held-out goal required fields that its training objective did not preserve, so closure alone does not rule out objective mismatch.

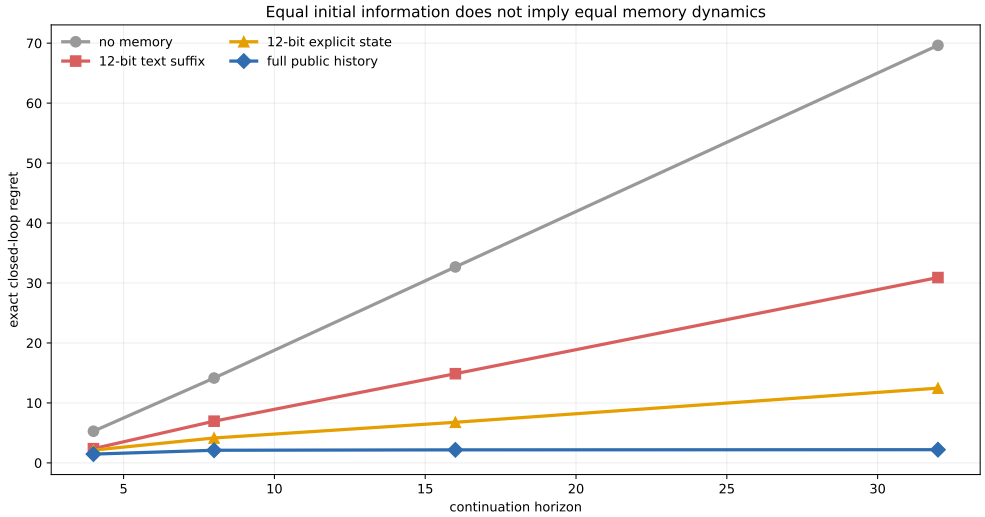


Figure 10: Exact closed-loop regret over continuation horizon.

Scientific conclusion. Equal snapshot information does not imply equal memory quality. A reusable state must summarize the current history well enough for the task, **and** remain sufficient after repeated updates. The relevant receiver therefore includes both the model that reads the state and the update rule that carries it forward.

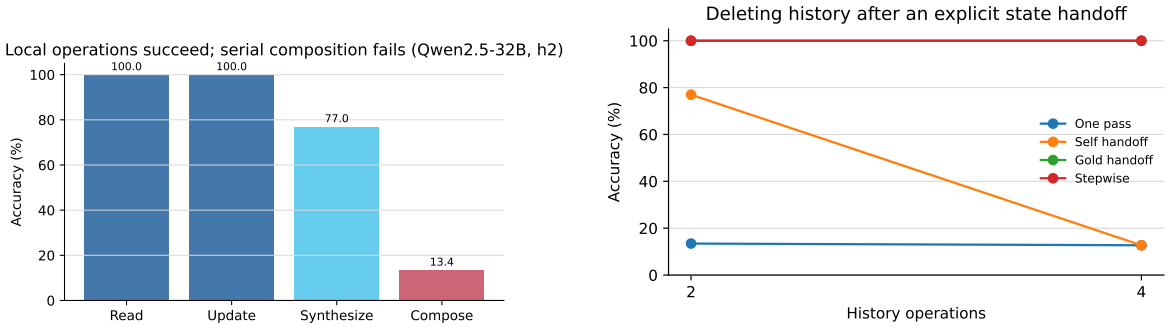
4.6 Native language-model state and explicit handoff

We designed a synthetic addition task where the necessary intermediate state is known exactly. A short program leads to one of eight possible three-bit states, and a separate pointer table maps that state to the final answer. This lets us test four abilities separately: read a state, update it, infer the state reached by a history, and use that inferred state in a final lookup.

The observation in Figure 11 is that frozen Qwen2.5-32B can read a supplied state and apply a local update perfectly. At horizon two it also infers the endpoint of the history with 76.98% accuracy. But when asked to infer that endpoint and immediately use it in the final lookup within one prompt, accuracy falls to 13.44%, close to the eight-state chance level.

If we stop after the model predicts the endpoint, delete the original history, and pass only that predicted state to a fresh model call, final accuracy returns to 76.98%. This matches endpoint-inference accuracy and therefore isolates the one-pass deficit to routing or composition rather than state inference. Passing the correct endpoint reaches 100%, as does explicit step-by-step decimal execution, ruling out the final lookup itself as the bottleneck.

Hidden-state probes also retain information about the path used to reach the endpoint, and the late-layer subspace we tested does not outperform a random subspace control. We therefore do not claim that the frozen model exposes one clean native code that can simply be patched between computations.



(a) Local competence with failed serial composition.

(b) Deleting history after an explicit handoff.

Figure 11: Frozen Qwen2.5-32B state-factorization assay.

4.7 Rate, shared meaning, and closure

This result ruled out easy reuse of an assumed native interface and led to a new question: what properties would make a supplied explicit interface work reliably?

To make the state space less trivial, we moved to a finite proof task. The state was in this case a four-bit fact ledger indicating which of four facts are currently true, giving 16 possible states. Horn-style implication rules updated the ledger one deduction at a time. At each step, the previous rules are discarded, and only the current state code is passed to the next model call, so the experiment would only be successful if the code itself carried the needed state.

We compared four code designs with equal training data and compute. A lossy three-bit code must merge some of the 16 states. A canonical code gives every state one stable name. A five-bit padded code adds unused capacity while keeping one name per state. Finally, a five-bit path-aliased code deliberately gives the same semantic state two possible names depending on how that state was reached. The last pair isolates extra capacity from path-dependent naming.

The observation in Figure 12 is that canonical and padded codes recover all 16 states in all three seeds and remain exact through the tested horizon of 256. Mean final accuracy is 97.22% for the canonical interface, compared with 72.22% for one-pass reasoning, a paired gain of 25 points with context-bootstrap interval $[12.50, 38.89]$. The three-bit code has one bit of unavoidable residual state uncertainty and never recovers the exact state. Extra unused capacity does not hurt the padded code, but path-dependent naming does: after three active deductions, the aliased code has only 30–40% exact state, versus 100% for the equally large padded code.

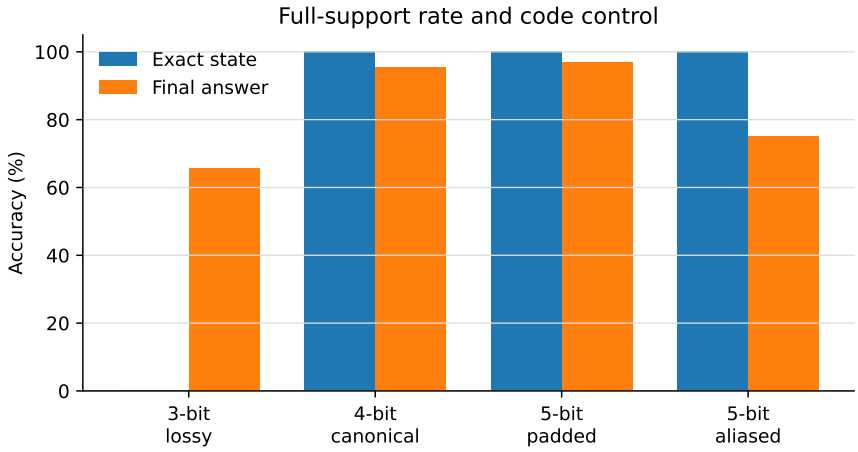


Figure 12: Full-support rate and code control in the later proof-state suite.

Scientific conclusion. The controls in Figure 12 separate three explanations. Lossy versus canonical tests insufficient capacity: the three-bit code cannot distinguish all 16 states, while the four-bit code can. Canonical versus padded shows that unused extra capacity is not harmful by itself; padded versus aliased holds the five-bit code size fixed and isolates the cost of path-dependent naming. Sufficient capacity is therefore necessary but not sufficient for a reusable state code.

A separate padding control rules out surface or context length alone and isolates the number of state-changing updates; we report it in Section A.6.

The proof task still had a very regular transition rule. We therefore asked whether the interface would survive a less regular rule containing several kinds of operation.

To test this, we moved to mixed register programs containing addition, XOR, swap, and conditional updates. We scored each local transition from the state actually supplied at that call, so an earlier mistake could not make a later correct update appear wrong. We also evaluated complete self-fed executions over 32 steps.

The model applies a single update correctly about 89% of the time. Yet Figure 13 shows that after roughly 31 self-fed updates, complete horizon-32 accuracy drops to only 7.92%, barely above the 6.25% chance rate; one-pass reasoning reaches 5.83%.

This gap separates local transition competence from accumulated reliability. The pattern was not model-dependent, and adding a fifth code bit did not repair it, ruling out model choice and one extra code bit as simple fixes within this test.

This failure marks the current limit of the interface approach, and more generally of forced causal state use. A transition function that is 89% correct in one step is not reliable enough for dozens of recursive uses, which RLMs easily reach in long deductions. Small semantic errors compound until the state is nearly useless. Long-horizon reliability therefore has to be measured under self-conditioned use, not inferred from gold-input one-step accuracy.

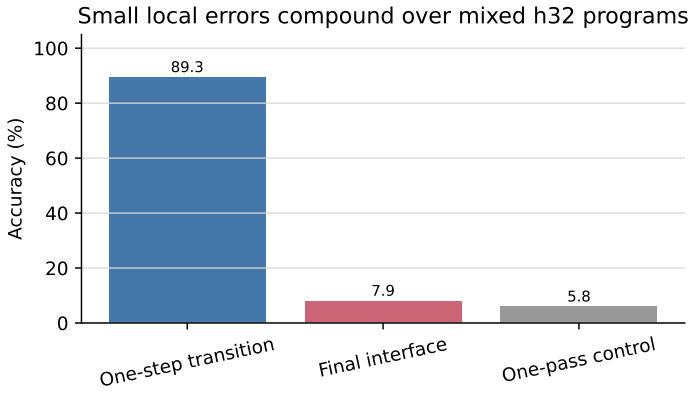


Figure 13: Local versus global reliability on held-out mixed register programs. Conditional transition accuracy is computed from the state actually supplied at each recursive call, so earlier errors do not make a later correct local update appear wrong. Roughly 89% local accuracy still yields 7.9% complete horizon-32 accuracy.

4.8 Takeaways

Our clearest results are that reasoning traces contain strong latent structure, but different objectives select different boundaries; that states can be easy to decode without being routed into the next computation; and that explicit compact interfaces work under controlled conditions only when they have enough capacity, one stable meaning per state, and sufficiently accurate self-updates.

The results also suggest a more continuous view of reasoning than a discrete, ordered structure of CoT steps. The next section presents a first attempt at unifying the two parts of the internship.

5 Synthesis and perspectives

5.1 A first bridge: reasoning and answer updates

To begin unifying the two tracks, we asked whether the reasoning chain and final answer require different amounts of update information. To test this hypothesis, we fine-tuned rank-16 all-linear LoRA adapters on MetaMathQA with two frozen NF4 backbones, Mistral-7B and Qwen2.5-7B. For each model, we trained on the full response, the reasoning span only, or the answer span only.

After training, each fixed adapter was compressed through the same dense codec ladder used in Section 3. We measured the exact serialized file rate in bits per LoRA value and the signed held-out bits per token saved against the frozen model, separately for the reasoning and answer spans. For a frozen model M and span utility U, we write

$$R^{\star}_{M,U}(\tau) = \min_{A} R(A) \quad \text{such that} \quad W_{M,U}(A) \geq \tau W_{M,U}(A_{\text{raw}}), \tag{4}$$

where $W_{M,U}$ is the held-out gain on that span.

The initial observation in Figure 14 is that, on the Mistral model, full-response and reasoning-only supervision save respectively 0.567 and 0.568 bits per reasoning token; on Qwen, they save 0.101 and 0.100. Adding answer supervision therefore has almost no effect on the gain over reasoning tokens. The compression rates, however, do not follow one ordering across models. At 90% retention, reasoning and answer require respectively 1.070 and 0.578 bits per value on Mistral, but 0.471 and 0.575 on Qwen. The claim that reasoning is more expensive than the answer therefore holds for Mistral, not for reasoning in general.

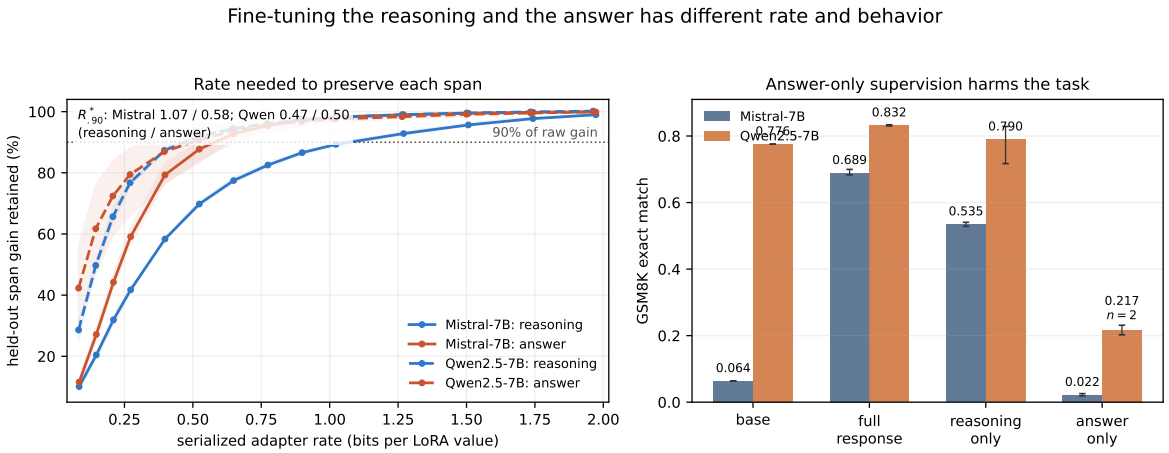


Figure 14: Fine-tuning rate and behavior for the MetaMathQA span-supervision study. Left: held-out gain retained by compressed full-response adapters, averaged over three seeds; shading shows the seed range. Right: mean GSM8K exact match for the raw adapters; error bars show the seed range.

5.2 What the two tracks establish together

This gives a completed reasoning-specific instance of the receiver dependence found in Section 3. A training corpus does not have one adapter rate by itself. The required rate depends on the frozen model and on which learned behavior must survive compression. It also matches the objective-relative result in Section 4: answer formation, symbolic updates, correctness, and continuation need not select the same representation. At training time, the carrier is a serialized adapter; at inference time, it is a trace partition or state code. In both cases, nominal size alone does not identify the information that supports a stated use.

We therefore have evidence for two utility-relative channels. In the training-time channel, the frozen model receives a compressed update and must retain a chosen behavior. In the inference-time channel, a later computation receives a compressed history or state and must retain a chosen use. The experiments show that both channels must be judged by what the receiver can still do.

5.3 What remains unresolved

The experiment above was still being analyzed when this report was written, and its results are preliminary. It uses one training corpus, two base models, one lexical split point, and far fewer held-out answer tokens than reasoning tokens. The experiment measures supervised next-token gain and GSM8K accuracy, so it does not establish how RLVR allocates update rate when only an outcome reward is supplied. Nor does it yet show whether a compressed adapter loses internal reasoning structure before or after it loses answer accuracy.

More broadly, the two halves of the report still meet at the level of a common question rather than one predictive law. Section 3 measures how many adapter bits preserve a behavior, while Section 4 shows that different future uses select different intermediate structure and that recursive use adds a closure requirement. We have not yet shown that a quantity measured before fine-tuning predicts a reasoning-specific rate, or that changing adapter rate predictably changes the internal state properties measured in the reasoning experiments.

5.4 How to test the bridge

The most promising next step is to make the two tracks predict each other. The correction-spectrum measure from Section 3 can be computed on the frozen model separately for reasoning

and answer tokens before training. If those utility-specific spectra predict the later $R^{\star}_{M,U}$ values across new models and datasets, the current model-relative account would become a predictive statement about which parts of a reasoning task are expensive to write.

Another path forward would be to compress a fixed reasoning adapter, and follow several properties along the same rate ladder. We could measure its objective-specific boundaries, operation decoding, trajectory geometry, and explicit state handoff tests from Section 4. If these properties fail at reproducibly different rates, we would have direct evidence that even within one trained model there is no single behavior-independent "amount of reasoning" stored in the update. If they fail together, that would instead identify a much stronger common bottleneck.

These two tests give us a concrete way to choose between the remaining hypotheses, rather than treating the shared interpretation as a result in itself.

6 Conclusion

The internship started from a broad question: how much information does a model have to acquire, or to keep, for a learned behavior to survive? The first obstacle was that "information" is not something one can simply go and measure. Much of the work consisted of replacing intuitive proxies with computable ones, e.g., a serialized adapter file, a held-out gain on a named span, a boundary set scored against a stated objective.

Our clearest results rule out simple explanations. Weight reconstruction error does not predict behavioral preservation. The code length of the source does not determine $R^{\star}$. Neither does the size of the improvement the fine-tune buys, since XBRL produced the largest gain of the campaign at the lowest measured rate, and the ordering of the five corpora rearranged itself as soon as the frozen base model changed. On the reasoning side, no single partition of the token lattice served answer formation, symbolic updates, correctness and compression at once. A state could be decoded cleanly and still not be routed into the next operation. And a transition rule that is right 89% of the time in one step becomes nearly useless when it is applied dozens of times in a single reasoning chain.

Research process. The internship was sustained practice in going from an observed phenomenon to a hypothesis, and then to an experiment designed to test it. Nearly every course correction in this report came from a result that did not work.

The results also support a common interpretation of the two tracks: useful information is not an intrinsic property of a dataset or a reasoning trace. It is relative to a receiver, a task, and, when reused recursively, an update rule. On the fine-tuning side, model-relative correction spectra are our strongest predictor of adapter rate; on the reasoning side, useful intermediate structure exists, but the structure we recover depends strongly on what it must be used for.

Future work. The next steps are to join the two lines more tightly, using the tests proposed in Section 5, and to extend the strongest results to new models and tasks. The fine-tuning track has since been developed into a paper that is currently under review.

I am very grateful to Paul Caillon and Alexandre Allauzen for their supervision, discussions, and freedom to pursue the questions that emerged throughout the internship.

A Additional experimental details and controls

A.1 Early adapter-factor asymmetry pilot

In an early pilot, A-only (with A being the down-projecting matrix of the LoRA adapter) LoRA training stayed near baseline, while B-only recovered much of full-LoRA performance in the tested setup. This functional asymmetry warned against treating equally sized parts of the adapter as equally valuable.

A.2 Prequential-code implementation

The conditional prequential code trains the model on earlier blocks and uses its log loss to encode later ones. A uniform fallback prevents a confidently wrong model from assigning an arbitrarily large code length, and a switch-code correction lets an entire block fall back to the shared base model when that is cheaper.

A.3 Codec ladder

At one bit and above, each LoRA matrix is quantized row by row. At one bit, a value is represented only by its sign and its row scale, so it reconstructs to $+\operatorname{mean}|w|$ or $-\operatorname{mean}|w|$. Higher rates use more quantization levels. Below one bit per value, we cannot give every value even one bit, so we instead keep only a fraction of LoRA rank directions, always keeping or dropping the corresponding directions in A and B together.

A.4 Exact diversity cells

Distinct examples	Epochs	$R^\star(.90)$	Seed range	Raw GSM8K EM
2,000	16	0.645	0.611–0.700	0.546
4,000	8	0.740	0.712–0.787	0.546
8,000	4	0.709	0.671–0.747	0.618
16,000	2	0.832	0.773–0.908	0.676
32,000	1	1.002	0.983–1.015	0.694

Table 3: $R^\star(.90)$ in serialized bits per raw adapter value, with optimizer steps and training-example presentations fixed.

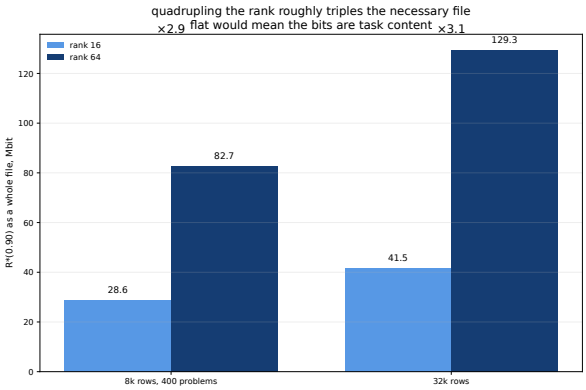
Task	Model	Groups	Mean $R^\star$, low $\rightarrow$ high	Positive seed pairs
SQL	Mistral-7B	10 $\rightarrow$ 100	0.375 $\rightarrow$ 0.406	2/3
SQL	Qwen2.5-7B	10 $\rightarrow$ 100	0.584 $\rightarrow$ 0.600	2/3
XBRL	Mistral-7B	10 $\rightarrow$ 30	0.260 $\rightarrow$ 0.318	3/3
XBRL	Qwen2.5-7B	10 $\rightarrow$ 30	0.397 $\rightarrow$ 0.469	3/3

Table 4: Fixed-example content-diversity replication. All 36 likelihood-based rate curves bracket $R^\star(.90)$.

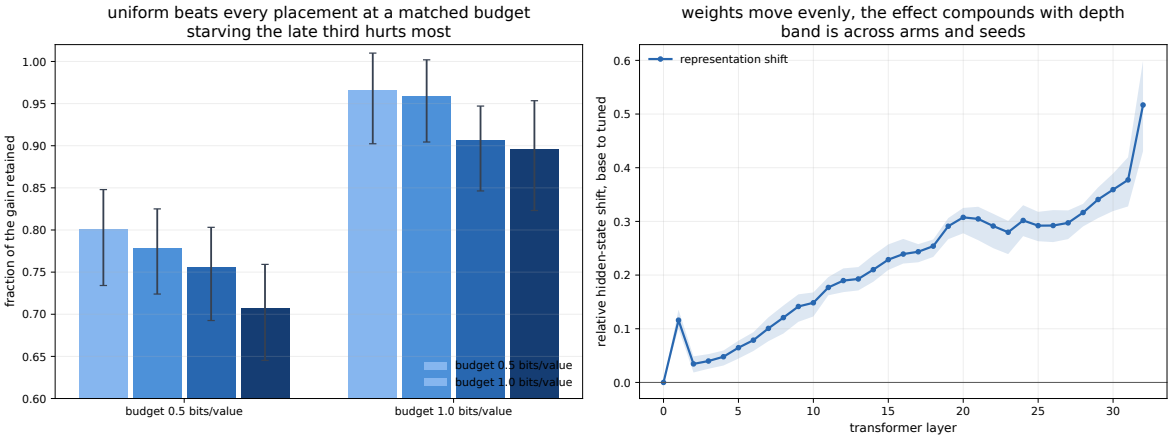
A.5 Adapter carrier and location controls

We tested whether the adapter architecture itself imposes a large cost independent of task content. As Figure 15 shows, quadrupling LoRA rank from 16 to 64 increases the whole-file $R^\star$ by roughly threefold in two matched conditions, suggesting that a substantial part of the message scales with the carrier. We also asked whether broad layer location could guide bit

allocation. At matched 0.5- and 1.0-bit budgets, uniform precision beats starving the early, middle, or late third of the network; starving late layers hurts most. Weight-update RMS is almost flat across depth, while the induced representation shift grows toward the final layers.



(a) Whole-file threshold at ranks 16 and 64.



(b) Layer-band allocation and depth-wise effect.

Figure 15: Container and location controls. Rank scaling is closer to proportional growth than to a fixed task message. At matched total rate, uniform precision beats starving any third of the network; late-layer starvation is most damaging even though update RMS is almost flat across bands.

The rank comparison changes the carrier while keeping the task fixed, ruling out the idea that $R^\star$ measures only a task message independent of adapter size. The matched-rate layer controls hold the total budget fixed and show that removing the same number of bits from different depth bands has different effects; the flat update RMS therefore cannot explain the late-layer sensitivity. These controls do not yet rule out a finer allocation within each band.

A.6 Surface length and active update depth

As Figure 16 shows, padding the rule stream at fixed active depth leaves accuracy unchanged, ruling out surface or context length alone. Increasing the number of state-changing deductions instead isolates active update depth.

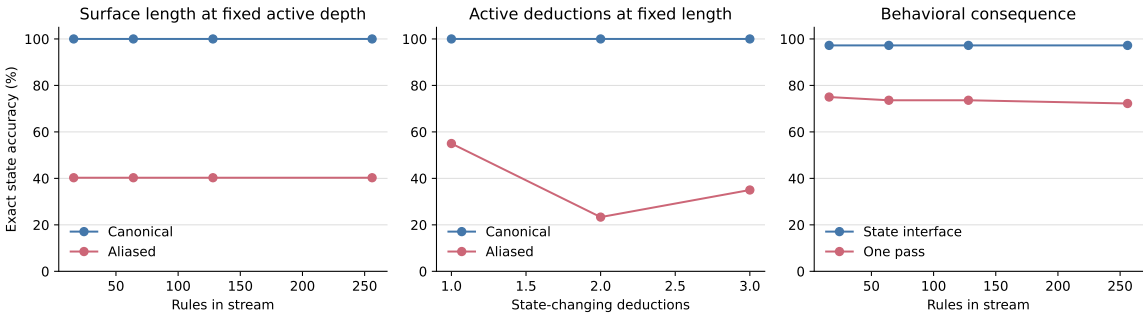


Figure 16: Length, active depth, and behavior in the canonical causal-state suite. Padding the rule stream at fixed active depth leaves exact state unchanged, whereas increasing state-changing deductions damages the aliased code. Canonical final behavior remains high through the padded horizons.

AI Usage Disclosure

Throughout the internship, OpenAI's harness Codex assisted me in the replication of existing papers, in first drafts of code implementations of well-defined research ideas and in debugging. AI output was not used to replace my judgment.

References

Aghajanyan, Armen, Luke Zettlemoyer, and Sonal Gupta (2021). "Intrinsic Dimensionality Explains the Effectiveness of Language Model Fine-Tuning". In: *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics*. DOI: 10.18653/v1/2021.acl-long.568.

Arora, Sanjeev et al. (2018). "Stronger Generalization Bounds for Deep Nets via a Compression Approach". In: *International Conference on Machine Learning*.

Berger, Toby (1971). *Rate Distortion Theory*. Prentice-Hall.

Bergsträßer, Pascal, Ryan Cotterell, and Anthony W. Lin (2026). "Transformers are Inherently Succinct". In: *International Conference on Learning Representations*. URL: https://openreview.net/forum?id=Yxz92UuPLQ.

Cover, Thomas M. and Joy A. Thomas (2006). *Elements of Information Theory*. 2nd ed. Wiley.

DeepSeek-AI (2025). "DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning". In: *arXiv preprint arXiv:2501.12948*. URL: https://arxiv.org/abs/2501.12948.

Dettmers, Tim et al. (2023). "QLoRA: Efficient Finetuning of Quantized LLMs". In: *Advances in Neural Information Processing Systems*.

Dutta, Subhabrata et al. (2024). "How to Think Step-by-Step: A Mechanistic Understanding of Chain-of-Thought Reasoning". In: *arXiv preprint arXiv:2402.18312*.

Fagnou, Erwan et al. (2024). "Chain and Causal Attention for Efficient Entity Tracking". In: *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*. DOI: 10.18653/v1/2024.emnlp-main.731.

Frantar, Elias et al. (2023). "GPTQ: Accurate Post-Training Quantization for Generative Pre-trained Transformers". In: *International Conference on Learning Representations*.

Grünwald, Peter and Teemu Roos (2019). "Minimum Description Length Revisited". In: *International Journal of Mathematics for Industry* 11.1, p. 1930001.

Hao, Shibo et al. (2025). "Training Large Language Models to Reason in a Continuous Latent Space". In: *Conference on Language Modeling*.

Hu, Edward J. et al. (2022). "LoRA: Low-Rank Adaptation of Large Language Models". In: *International Conference on Learning Representations*.

Li, Yixiao et al. (2024). "LoftQ: LoRA-Fine-Tuning-Aware Quantization for Large Language Models". In: *International Conference on Learning Representations*.

Lin, Ji et al. (2024). "AWQ: Activation-aware Weight Quantization for LLM Compression and Acceleration". In: *Proceedings of Machine Learning and Systems*.

Liu, Zechun et al. (2025). "ParetoQ: Improving Scaling Laws in Extremely Low-bit LLM Quantization". In: *Advances in Neural Information Processing Systems*. Vol. 38. DOI: 10.52202/085713-3055.

Lotfi, Sanae et al. (2022). "PAC-Bayes Compression Bounds So Tight That They Can Explain Generalization". In: *Advances in Neural Information Processing Systems*.

Qian, Chen et al. (2025). "Demystifying Reasoning Dynamics with Mutual Information: Thinking Tokens are Information Peaks in LLM Reasoning". In: *Advances in Neural Information Processing Systems*.

Shao, Zhihong et al. (2024). "DeepSeekMath: Pushing the Limits of Mathematical Reasoning in Open Language Models". In: *arXiv preprint arXiv:2402.03300*. URL: https://arxiv.org/abs/2402.03300.

Sun, Lihao et al. (2026). "LLM Reasoning as Trajectories: Step-Specific Representation Geometry and Correctness Signals". In: *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics*. DOI: 10.18653/v1/2026.acl-long.1237.

Vaswani, Ashish et al. (2017). "Attention Is All You Need". In: *Advances in Neural Information Processing Systems*. URL: https://arxiv.org/abs/1706.03762.

Wei, Jason et al. (2022). "Chain-of-Thought Prompting Elicits Reasoning in Large Language Models". In: *Advances in Neural Information Processing Systems*.

Xiao, Guangxuan et al. (2023). "SmoothQuant: Accurate and Efficient Post-Training Quantization for Large Language Models". In: *International Conference on Machine Learning*.

Xu, Yuhui et al. (2024). "QA-LoRA: Quantization-Aware Low-Rank Adaptation of Large Language Models". In: *International Conference on Learning Representations*.

Yang, Yukang et al. (2025). "Emergent Symbolic Mechanisms Support Abstract Reasoning in Large Language Models". In: *International Conference on Machine Learning*.

Yao, Shunyu et al. (2023). "ReAct: Synergizing Reasoning and Acting in Language Models". In: *International Conference on Learning Representations*.

Yao, Zhewei et al. (2022). "ZeroQuant: Efficient and Affordable Post-Training Quantization for Large-Scale Transformers". In: *Advances in Neural Information Processing Systems*.

Young, Sean I. (2025). "Radio: Rate–Distortion Optimization for Large Language Model Compression". In: *Proceedings of the 42nd International Conference on Machine Learning*. Vol. 267. Proceedings of Machine Learning Research. PMLR, pp. 72819–72836. URL: https://proceedings.mlr.press/v267/young25b.html.

Yu, Sheldon et al. (2025). "Explainable Chain-of-Thought Reasoning: An Empirical Analysis on State-Aware Reasoning Dynamics". In: *Findings of the Association for Computational Linguistics: EMNLP 2025*. DOI: 10.18653/v1/2025.findings-emnlp.904.

Zhang, Qingru et al. (2023). "AdaLoRA: Adaptive Budget Allocation for Parameter-Efficient Fine-Tuning". In: *International Conference on Learning Representations*.

Zhang, Yuyi et al. (2025). "Do Latent Tokens Think? A Causal and Adversarial Analysis of Chain-of-Continuous-Thought". In: *arXiv preprint arXiv:2512.21711*.

Zhou, Wenda et al. (2019). "Non-Vacuous Generalization Bounds at the ImageNet Scale: A PAC-Bayesian Compression Approach". In: *International Conference on Learning Representations*.

Zhu, Jiacheng et al. (2024). "Asymmetry in Low-Rank Adapters of Foundation Models". In: *arXiv preprint arXiv:2402.16842*.