LANGUAGE MODELS AS INFORMATION RATCHETS: AN INFORMATION-THERMODYNAMIC PERSPECTIVE ON IN-CONTEXT LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

In-context learning raises two questions usually studied apart: how much a prompt can teach a model, and how much task diversity pretraining needs before a model learns in context rather than memorizing. Reading a language model as an information ratchet—a machine that turns predictions into work—makes one quantity answer both: the mutual information between task and context, the learning toll. The toll is a floor no model can beat. It is tight for transformers trained on a fresh task in every sequence, so their whole loss curve follows from the task family alone; frozen language models pay several times it, and what they overpay is a failure to keep learning, not a wrong prior. The toll also counts tasks: a context tells apart only about e^{toll} of them, so beyond that memorizing stops paying. Trained transformers switch where it says, until the pool outgrows what they can hold, and that point is set by the task family and its prior, not by the number of a task's parameters. Post-training a pretrained model shows the same selection.

1 INTRODUCTION

A language model that has finished training can still learn. Shown a handful of examples of a task it has never seen, it predicts the next answer more accurately, and it does so without changing a single weight (Brown et al., 2020). This ability, called in-context learning, is learning of an unusual kind: the model's parameters stay fixed, and everything it learns about the new task is held in its reading of the prompt. Controlled studies show that it does not always appear: a model pretrained on a small pool of tasks memorizes them, while one pretrained on a large pool learns each new task from its context (Raventós et al., 2023; Kirsch et al., 2022). Two questions follow. *(1) How much can a context improve a model's predictions on a new task?* And *(2) when does pretraining produce a model that learns each task from its context rather than one that memorizes its training tasks?*

Information-theoretic and Bayesian-based theories answer the first question. They treat in-context learning as inference of a latent task and give the loss of an ideal learner exactly (Xie et al., 2022; Jeon et al., 2024; Riechers et al., 2025), and pretrained models have been compared with such ideal predictors on synthetic sequences (Falck et al., 2024; Chlon et al., 2025; Ozelim & Zenil, 2026) to measure the gap between what a context allows a model to learn and what the model actually learns. Accounts of emergence answer the second question. Empirical studies established that pretraining switches from memorizing to learning as the pool of tasks grows, and that the switch depends on the statistics of the training data (Chan et al., 2022; Raventós et al., 2023). Learning dynamical and rational analyses explain why it occurs, as a race between two circuits during training (Singh et al., 2023; Nguyen & Reddy, 2025).

Despite this progress, the two questions are usually answered in separate frameworks and in different terms. A few accounts do answer both, but each of them has the limitations, e.g., the analysis can't be done before training (Wurgaft et al., 2025), or only solvable on linear transformers (Lu et al., 2025; Takanami et al., 2026; Kim et al., 2026), a simplified transformer for theoretical analysis. What is missing is a single quantity to answer these two questions, which is computable from the task family before model training and is applicable for the real transformer models.

Maxwell's demon gives both questions one language, in which information is a thermodynamic resource: a belief about which of A compartments holds a molecule pays work that grows with

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

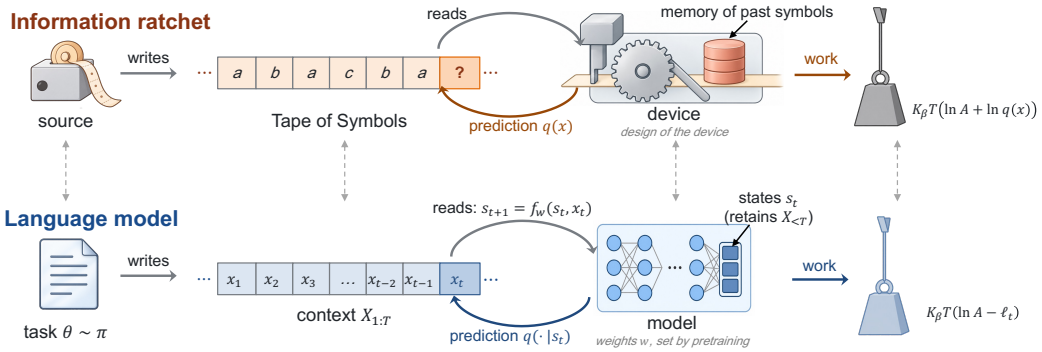

Figure 1: **A language model reading its context is an information ratchet.** The ratchet (top) and the LLM (bottom) have clear matching: the source that writes the tape is the task in LLM, the tape is the context, and the machine's belief about the next compartment is the model's next-token distribution $q(\cdot \mid s_t)$. Read this way, each token the model predicts is worth $k_B T_{\mathrm{B}}(\ln A + \ln q(x))$ of work to the ratchet: the most the alphabet allows, less the model's loss on that token.

the logarithm of the probability it gave the compartment the molecule was in (Szilard, 1929; Kelly, 1956). A language model is a modern form of the Maxwell's demon, the information ratchet, which reads a tape of symbols and extracts work from each symbol by predicting it from the symbols it has already read (Mandal & Jarzynski, 2012; Boyd et al., 2016). The work it extracts depends only on how well its predictions match the tape, so the device whose predictions match best extracts the most (Kelly, 1956; Boyd et al., 2022). Thus, the next-token predictions in LLMs play the role of the device's predictions, the better it predicts the more work it would extract, and model pretraining, by minimizing its prediction loss, selects the device that extracts the most work. In this language both questions turn on one quantity, how much a tape shows about its source: *question (1)* asks what a device gives up when it must infer that source from the symbols it has already read instead of being told it, and *question (2)* asks when the sources seen in training grow too many for a tape to tell apart, so that holding them in memory stops paying.

We call this number the *learning toll*. It answers the first question directly: no model can lose less than the toll, whatever its size or however it was trained. Transformers trained on a fresh task in every sequence lose exactly the toll, so the toll predicts their whole loss curve before any of them is trained, while frozen language models lose several times more, and the excess is what the context offered them and they failed to use (§3). It answers the second through a count: a context tells apart only about e^{toll} tasks, so once pretraining holds more tasks than that, memorizing them buys nothing and pretraining produces an in-context learner instead. Trained transformers switch where this count says, across task families, context lengths and priors, so long as they can hold the pool it asks for; where they cannot, they switch at what they can hold instead (§4).

2 From information ratchet to in-context learning

Maxwell's demon turns predictions into work. A demon that learns which half of a cylinder at temperature T_{B} holds a molecule can let the partition slide and collect $k_B T_{\mathrm{B}} \ln 2$ of work from the bath (Szilard, 1929), and erasing that record costs at least as much (Landauer, 1961; Bennett, 1982): one bit and $k_B T_{\mathrm{B}} \ln 2$ are exchangeable. The same holds for partial knowledge. With A equal compartments and a belief q over them, the demon sets the walls in proportion to q and collects

$$W = k_B T_{\mathrm{B}} \big(\ln A + \ln q(x) \big) \tag{1}$$

when the molecule is in compartment x (Appendix D.1). Only the probability it gave to what happened matters: certain and right collects $k_B T_{\mathrm{B}} \ln A$, undecided collects nothing, confidently wrong pays (Kelly, 1956; Vinkler et al., 2016). An information ratchet repeats this exchange along a tape of compartments that pass one at a time, setting the walls for each from what it has kept of those already gone (Mandal & Jarzynski, 2012; Boyd et al., 2016). Two things follow from equation 1: the machine that predicts best collects the most work, so selecting for work is selecting the best model of the tape (Boyd et al., 2022); and a regularity spanning many compartments is available only to a machine whose state keeps enough of the past (Boyd et al., 2017b; Still et al., 2012).

A language model is an information ratchet. An LLM reads $x_1, \ldots, x_T$ over a vocabulary of A tokens and factorizes

$$p(x_{1:T}) = \prod_{t=1}^{T} q(x_t \mid s_t), \qquad s_{t+1} = f_w(s_t, x_t), \tag{2}$$

with s_t what the model retains of the tokens before t, f_w the update set by the weights w, and $q(\cdot \mid s_t)$ the next-token distribution. This is a ratchet: the tokens are the tape, s_t is what the machine carries from the compartments, and $q(\cdot \mid s_t)$ is the belief it holds before the next one arrives (Figure 1). By equation 1 it extracts $k_B T_{\mathrm{B}}(\ln A - \ell_t)$ at position t, where $\ell_t = \mathbb{E}[-\ln q(X_t \mid s_t)]$ is the loss.

In-context learning is the fall of this loss with the weights fixed, and every nat of it is worth another $k_B T_{\mathrm{B}}$ of work. A task $\theta \sim \pi$ generates examples $X_{1:T} \sim P_\theta$, and positions index examples throughout—one token in most of our families, otherwise ℓ_t is the loss on the whole example—so the first $t-1$ of them teach the model about the task behind X_t. The loss splits three ways:

$$\ell_t = \underbrace{H(X_t \mid X_{<t}, \theta)}_{\text{loss if the task were known}} + \underbrace{I(\theta; X_t \mid X_{<t})}_{i_t\text{: task not yet learned}} + \underbrace{\mathbb{E}\, D_{\mathrm{KL}}(p_t \,\|\, q_t)}_{d_t\text{: dissipation}}. \tag{3}$$

The first term is what a model told the task, i.e., the *oracle model*, would still pay. The second, i_t, is what an *ideal in-context learner* pays on top for not yet knowing the task, and it shrinks as examples accumulate. The third is what an LLM q_t loses beyond that ideal in-context learner p_t defined by the expected divergence between them. We call it *dissipation*, because in the ratchet it is the work lost by setting the walls from a mismatched belief (Kolchinsky & Wolpert, 2017).

Definition of the analysis instruments. Built from i_t and d_t, we define

$$\underbrace{\Lambda(T) = \sum_{t=1}^{T} i_t = I(\theta; X_{1:T})}_{\text{learning toll}}, \qquad \underbrace{g_t = i_1 - i_t}_{\text{learning gain}}, \qquad \underbrace{\eta_t = 1 - \frac{d_t}{g_t}}_{\text{learning efficiency}}. \tag{4}$$

As illustrated in Figure 2, the *learning toll* is what the ideal in-context learner pays above the oracle model over a whole context of length T for having to read the task from it; it is the mutual information between the task and the context. The *learning gain* is the part of i_1 the context has provided by demonstration t, since $i_t + g_t = i_1$: the ideal learner improves by exactly g_t, and any other model improves by g_t less whatever its own dissipation has grown by. The *learning efficiency* is the share of that gain a model has taken, measured from the ideal learner's own starting point i_1: a model paying $i_t + d_t$ above the oracle stands $g_t - d_t$ below i_1, against the ideal learner's g_t. It is 1 for the ideal learner, 0 for a predictor that never reads the context—which pays exactly $d_t = g_t$—and negative for one the context leads astray, which ends worse off than if it had never read it. A model that starts from a worse belief than the ideal learner is charged for that too, since its d_1 is already part of d_t.

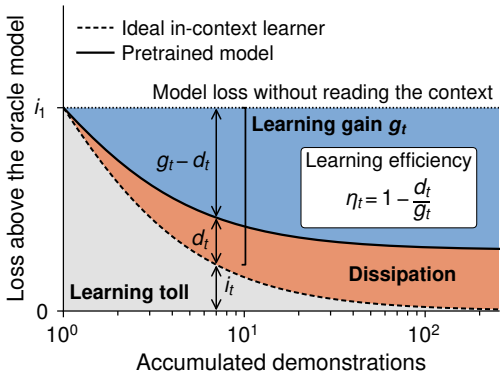

Figure 2: **Illustration of the instruments.**

3 The learning toll bounds in-context learning

How much a context can improve a model's predictions on a task it has never seen? The learning gain answers it, and the learning toll bounds what any model can do with it.

Theorem 1 (The learning toll is the limit of in-context learning)**.** *Let a task $\theta \sim \pi$ generate $X_{1:T}$, and let an LLM with weights independent of θ predict each X_t from $X_{<t}$. At every position $d_t \geq 0$, with equality if and only if the LLM's prediction equals that of the ideal in-context learner. Thus,*

$$\sum_{t=1}^{T} \Big(\ell_t - H(X_t \mid X_{<t}, \theta)\Big) \;=\; \Lambda(T) + \sum_{t=1}^{T} d_t. \tag{5}$$

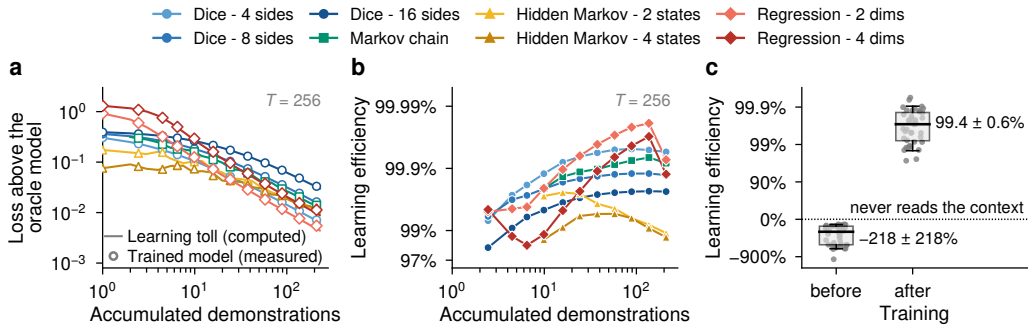

Figure 3: **The learning toll predicts what a trained transformer loses.** Transformers trained on a fresh task in every sequence on eight task families (Figure 11 for details). **a**, The toll paid at each demonstration, i_t, computed from the family alone (lines) against the same quantity measured from the trained model, $i_t{+}d_t$ (points). **b**, The same models in learning efficiency: at every demonstration, the share of the learning gain provided so far that the model has taken. **c**, Learning efficiency before and after training; boxes span the quartiles, whiskers the 5th to 95th percentiles.

No such LLM loses less than the oracle model plus the learning toll, or extracts more than the oracle model's work minus $k_B T_{\mathrm{B}}\,\Lambda(T)$, and the limit is reached if and only if $d_t = 0$ after every demonstration.

Proposition 1 (Size of the learning toll)**.** *If π is supported on K tasks, $\Lambda(T) \le \ln K$; if, in addition, examples are i.i.d. given the task and the K example distributions are distinct, i_t decays exponentially in t. If examples are i.i.d. given the task and the tasks form a smooth* dim*-parameter family, with a positive continuous prior density and a nonsingular Fisher information, then $i_t = \frac{\dim}{2t}\,(1 + o(1))$ and $\Lambda(T) = \frac{\dim}{2}\ln T + c_\pi + o(1)$, where c_π depends only on π.*

Proofs of Theorem 1 and Proposition 1 are in Appendix B. The statement is asymptotic: it fixes how the toll grows, but not its value at a given length. At the lengths typical of in-context learning the neglected terms are not small (Appendix D.2), so we compute the toll of each task directly and use these values, rather than the asymptotic forms, in all comparisons below.

3.1 The toll predicts the whole loss curve before training

Minimizing the loss over training sequences whose task is drawn afresh each time has a single optimum: the prediction that averages over the tasks still consistent with what has been read, that is, the ideal in-context learner. The toll therefore gives more than a floor. It gives, before any training and from the task family alone, the loss of a well-trained model after every demonstration: the oracle loss plus the toll. In this section, we test if the trained model can really achieve this optimum. We train one transformer for each of eight task families at five context lengths, under one protocol throughout: dice over 4, 8 and 16 symbols, a Markov chain, hidden-Markov families with two and four hidden states, and linear regression in 2 and 4 dimensions, at $T = 16$ to 256 demonstrations. A *demonstration* is one position of a context: a symbol in the discrete families, an input–output pair in regression. A fresh task is drawn for every training sequence, so no task recurs and none can be memorised. Appendix C gives the details on the task families, architecture, and protocol.

The theoretical prediction holds by the empirical validation. After every demonstration the trained model's loss above the oracle model lies on the toll computed before training, in all eight families (Figure 3a; without the oracle model subtracted away, Figure 11): over the 256 demonstrations of the longest context the median ratio of the two is 1.01 to 1.07. Read as learning efficiency the agreement holds at every point of a context and not only at its end, between 98.1% and 99.98% of what the context has provided so far (Figure 3b); at the end of the context the models take 97.3% to 99.94% of the learning gain in all settings, $99.4 \pm 0.6\%$ on average (Figure 3c; Figure 9 for full results). Before any training every one of the networks ends its context worse off than a predictor that never reads it, taking $-218 \pm 218\%$ of the gain on average, so training carries every setting from less than nothing to nearly all of what the context provides (Figure 10 for full results). These models also carry the ideal learner's posterior linearly in their residual streams (Figure 12).

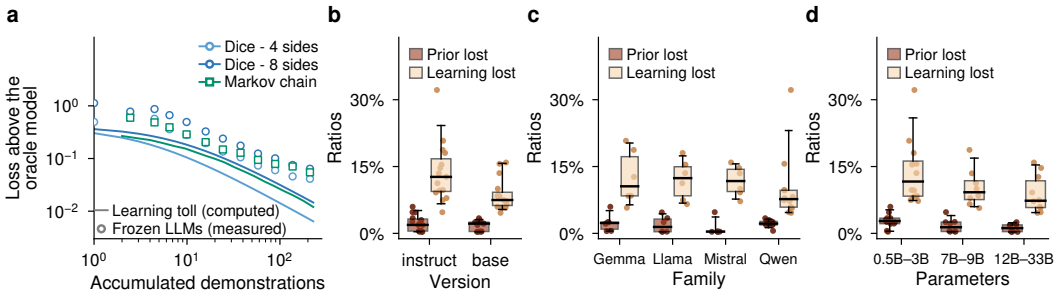

Figure 4: **Frozen language models pay several times the learning toll, and most of what they lose is learning, not the prior.** Thirty open models—the base and instruction-tuned releases of Qwen2.5, Llama, Gemma-2 and Mistral, 0.5 to 33 billion parameters—on three task families at $T = 256$. **a**, The learning toll computed from the family alone (lines) against the same models' measured loss (markers); the gap between the two is their dissipation. **b–d**, Their dissipation, split into the part a wrong prior explains (prior lost) and the part it does not (learning lost), each as a share of the learning gain, by model version (**b**), model family (**c**) and scale (**d**).

The learning toll is a prediction, not only a bound.
What a context can improve a model's predictions by—the learning gain—is fixed by the task family before any training. A transformer trained on a fresh task every time takes 97.3 to 99.9% of it across eight task families and five context lengths, whereas the same network before training ends its context worse off than one that never reads it.

3.2 Frozen LLMs take most of what a context offers, and lose the rest to learning rather than to their prior

The learning gain is computed from the task alone, so it can be turned on models we did not train, to ask how much of what a context provides a pretrained language model takes and where the rest goes. We recorded the predictions of thirty open models—the base and instruction-tuned releases of Qwen2.5, Llama, Gemma-2 and Mistral, from 0.5 to 33 billion parameters—on three of the task families at $T = 256$: dice - 4 sides, dice - 8 sides and the Markov chain. Figure 4a sets what these models pay above the oracle model beside the learning toll computed from the family alone, and the gap between the two is their dissipation. The curves are closest at the first demonstrations and separate as the context grows: summed over the whole context the average model pays 2.6 to 3.4 times the toll, and over the last eighth of it 4.3 to 6.4 times (Figure 14), so that more than three quarters of what it still pays there is waste rather than the price of learning the task. These models do read their context, and they take most of what it offers—the median base model ends with 90% of the learning gain (Figure 13)—but they stop improving while an ideal learner is still improving.

To this end, we analyze the dissipation $\mathbb{E}\, D_{\mathrm{KL}}(p_t \,\|\, q_t)$, the divergence between the ideal in-context learner and the model. A model can stand away from that ideal in two ways: by starting from a different belief about the task, or by updating differently as demonstrations arrive. The first is the more forgiving failure, since the evidence washes a starting belief out as the context grows, while a model that updates wrongly keeps paying. To tell them apart we fit to each model the closest add-α rule—the ideal learner of a $\mathrm{Dir}(\alpha)$ prior rather than the true one, that is, add-α smoothing of the running counts—on demonstrations 2 to 128 and write p_t^{α} for its prediction. What that rule itself dissipates, $d_t^{\mathrm{prior}} = \mathbb{E}\, D_{\mathrm{KL}}(p_t \,\|\, p_t^{\alpha})$, is the part of the model's dissipation that a wrong starting belief accounts for. The learning gain then falls in three parts,

$$g_T = \underbrace{(g_T - d_T)}_{\text{taken}} + \underbrace{d_T^{\mathrm{prior}}}_{\text{prior lost}} + \underbrace{d_T - d_T^{\mathrm{prior}}}_{\text{learning lost}}. \tag{6}$$

Because the add-α rules are only one direction in which a starting belief can be wrong, the prior lost is a lower bound on what a wrong prior costs, and the learning lost an upper bound on what remains.

Read through this split, what these models leave is not a prior that the evidence would wash out but a failure to keep learning from it (Figure 4b–d). However the models are grouped, the prior lost is

a median 3% of the learning gain or less at the end of the context, and the learning lost between 7 and 13%. Both fall with scale, and they fall together: the median model gives up 13.3% of the gain below 3B, 10.7% at 7–9B and 8.8% above 12B. Instruction tuning moves the total the other way, and moves only the learning part: the median rises from 10.0% for the base models to 13.2% for their instruction-tuned versions, while the prior lost stays at 2.2 and 1.9%, and in 14 of the 15 pairs the instruction-tuned model gives up more of the gain than its own base. Against these two the model families differ little, from 10.0% for Qwen2.5 to 12.8% for Llama and Mistral, and only Qwen2.5 holds its place from one task to the next (Figure 15).

What frozen LLMs leave is learning lost, not a wrong prior.
At the end of a context of 256 demonstrations the median base model has taken 90% of the learning gain, but it paid several times the learning toll to get there, and what it left is almost all learning lost: however the models are grouped, the prior accounts for 3% of the gain or less. The shortfall shrinks with scale and grows with instruction tuning, which moves the learning part and leaves the prior where it was.

4 THE LEARNING TOLL LOCATES THE SWITCH FROM MEMORIZING TO LEARNING

Pretraining does not always produce a model that learns from its context: when its training pool holds few tasks a model memorizes them, and only past some pool size does it infer each new task instead (Raventós et al., 2023). The learning toll fixes that size, because the nats a context spends on learning also count the tasks it can tell apart, of order $e^{\Lambda(T)}$ of them.

Given a pool $\mathcal{K}$ of K tasks drawn from π, each pretraining sequence taking its task uniformly from the pool (Raventós et al., 2023; Wurgaft et al., 2025), two ideal predictors bracket what pretraining can produce. The *ideal memorizer* carries $\mathcal{K}$ in its weights and predicts with the posterior predictive over it, so a context need only tell it which member it faces; the *ideal in-context learner* carries only π and has to read the task itself. Their learning tolls are $\Lambda_K(T)$ and $\Lambda_\pi(T)$. On pretraining sequences—all that pretraining can see—the memorizer loses $\Delta(K,T) = \Lambda_\pi(T) - \mathbb{E}[\Lambda_K(T)] \geq 0$ nats less, and this *retrieval margin* is what storing the pool is worth. Telling K stored tasks apart costs at most $\ln K$ nats, so the margin vanishes once the pool outgrows what a context can tell apart (Appendix B): beyond that point the two predictors are indistinguishable where selection acts, and any cost of storage selects the learner. Write p_t^K for the memorizer's prediction, as p_t is the ideal learner's in equation 3. A trained model q_t is placed between the two by its distance to the memorizer over a whole context, $d_{\mathrm{mem}}(K,T) = \sum_{t=1}^{T} \mathbb{E}\, D_{\mathrm{KL}}\big(p_t^K \,\|\, q_t\big)$, in the nats of the margin, and by the fraction of the margin it has covered, $\varphi = d_{\mathrm{mem}}/\Delta$: at $\varphi = 0$ the model is the memorizer, at $\varphi = 1$ it stands as far from the memorizer as the ideal learner does. The pool at which φ passes $\frac{1}{2}$ is the number of tasks the model holds, and $\Delta - d_{\mathrm{mem}}$ is what the pool still saves it.

Theorem 2 (Selection threshold)**.** *Let storing the pool cost $c_{\mathrm{pen}} > 0$ nats, and let pretraining memorize when $\Delta(K,T) > c_{\mathrm{pen}}$. $\Delta(K,T)$ is nonincreasing in K, so a single switch $K^*(T) = \min\{K : \Delta(K,T) \leq c_{\mathrm{pen}}\}$ separates the two regimes, with $\ln K^*(T) \geq \Lambda_\pi(T) - c_{\mathrm{pen}}$ for every family and $\ln K^*(T) = \Lambda_\pi(T) + O(1)$ for categorical tasks under a uniform Dirichlet prior.*

Proofs and the two constants are in Appendix B. We fix c_{pen} at one nat per context throughout.

4.1 THE SWITCH POINT IS COMPUTABLE BEFORE TRAINING

Transformers trained on pools of $K = 2$ to 131,072 tasks from three families at $T = 16$–512 stop memorizing where the toll says (Figure 5a). Each switch is read where storing the pool stops saving a nat: the margin for the exact pair, that margin less the model's own distance to the memorizer for a trained one. Over the ten settings in which the toll is the binding limit both readings are lines in the toll—slope 1.04 computed and 0.94 measured, $R^2 \geq 0.99$, over switches from 4 to 119 tasks.

What a model holds is a second limit that no computation from the task family can know (where the model capability is analyzed in Figure 16). Across the sixteen settings it is 100 to 470 tasks (Figure 5b), and that level belongs to the training budget those settings share rather than to the size of the network: four times the steps raises it to 646 (§4.2). Where the toll asks for fewer tasks, measured and computed values agree to within 0.34 nats in nine of ten. Where the toll asks for more

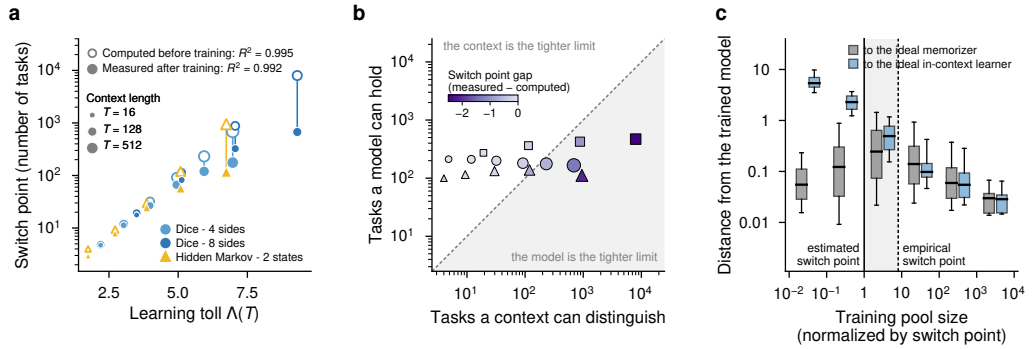

Figure 5: **The toll computes the switch; a trained model leaves the memorizer for the learner over the decade of pool size above it.** **a**, The pool at the switch against the learning toll: open marker is computed before training; filled marker is measured after it. **b**, Tasks that a context can distinguish against tasks that a model holds, coloured by switch point gap between the measured and computed ones. **c**, Distance from each trained model to the two ideal predictors, against the pool in units of that setting's switch.

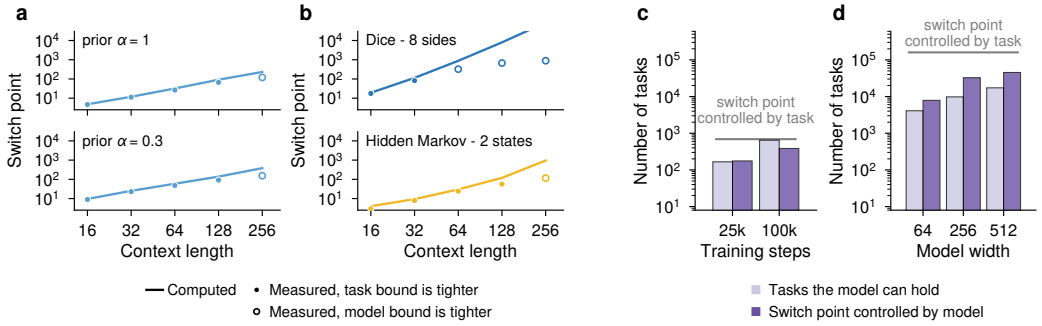

Figure 6: **Both data and model can move the switch point.** **a**, The four-sided die under two priors and **b**, two task families of almost the same size: lines are the switch computed before training, points the switch measured after it, filled where the task bound is tighter and open where the model bound is. **c**, **d**, The same transformer trained four times longer, and widened: light bars are the tasks it holds, dark bars its measured switch, the grey line the computed switch.

tasks than the model holds, the measured switch leaves the computed one by 0.66 to 4.56 nats and lands instead within a factor of two of what the model holds—0.7 to 1.9 times it in the six settings concerned.

What the theory places at a point, training spreads over a bit (Figure 5c). A tenth of the way to the switch a trained model is the exact memorizer of its pool, 0.055 nats from it against 5.4 from the ideal learner. The switch is where the two ideal predictors come within a nat of each other, so it opens a crossover rather than marking a jump: at the switch the model is still a quarter of a nat from the memorizer and half from the learner, and the two distances cross only at 8.1 times it, at a few hundred to a few thousand tasks. Beyond it the ideal pair converges and the distinction dissolves, as the resolvability bound of Lemma 6 in Appendix B requires.

The toll computes the switch, and a trained model leaves the memorizer a decade later.
One number fixed before training says where memorizing stops paying, and trained models follow it wherever they can hold the pool it asks for: slope 0.94 measured against 1.04 computed, $R^2 \geq 0.99$. Below the switch a model is the exact memorizer of its pool; it becomes the ideal learner within the next decade, at the few hundred tasks this training budget lets it hold.

4.2 Both data and model can move the switch point

Theorem 2 fixes one limit on the switch, the pool $K^*(T)$ that the toll computes, and §4.1 measured a second, the number of tasks K_{hold} a trained model holds. The tighter of the two decides where a

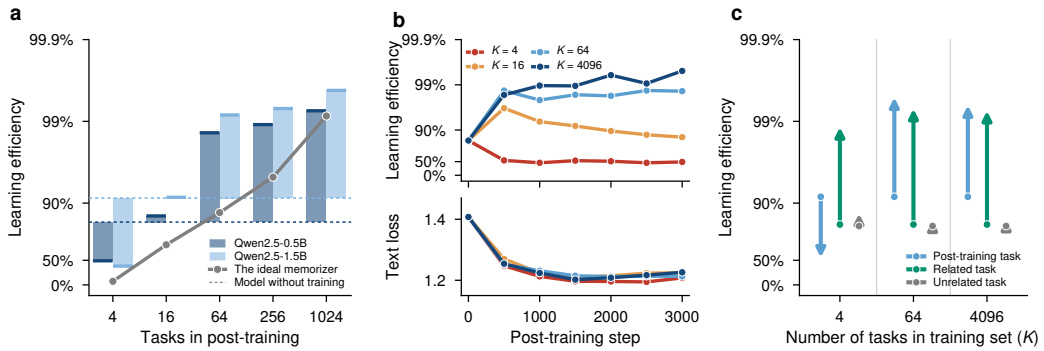

Figure 7: **Post-training decides which side of the switch a pretrained model ends on. a**, Each bar runs from the model before post-training to the model after it; grey line, the exact memorizer of each pool. **b**, The same runs along post-training: above, the learning efficiency on a new task; below, the held-out text loss. The pool size changes the in-context learning capability, not the text modeling. **c**, Qwen2.5-1.5B post-trained on two task families at once, with a third it never sees

trained model switches. Each limit has its own levers, and we evaluate the impact of the data side (Figure 6a,b) and the model side (Figure 6c,d).

The data move the first limit. Under two priors on one task family (dice - 4 sides) the number of a task's parameters is identical but the switch is not: a $\mathrm{Dir}(0.3)$ pool needs about twice as many tasks as a $\mathrm{Dir}(1)$ one before memorizing stops paying, and each prior's measured switch follows its own computed line (Figure 6a). Between two task families of almost the same size, $\mathrm{dim}{=}7$ and $\mathrm{dim}{=}8$, the computed switch grows at 2.76 and 1.45 per e-fold of context and the trained models at 2.10 and 1.53 (Figure 6b). A count of parameters predicts neither contrast.

The model moves the second. With the data fixed the computed line cannot move, and what moves is how many tasks the model holds. Training the same transformer four times longer raises that from 166 to 646 tasks, and its measured switch rises with it from 176 to 385, against a computed limit of 689 that stays where it was (Figure 6c). Where that limit lies above every pool we ran, so that the model binds throughout, widening the transformer from 64 to 512 raises its measured switch from 7.9k to 45k tasks (Figure 6d): 0.29 nats per doubling of parameters, against the 0.69 that storing a fixed number of tasks per parameter would require.

The data move what a context can distinguish; the model moves what it can hold.
Two task families of almost the same size grow at 2.76 and 1.45 per e-fold of context, and one task family under two priors switches at levels twice apart: what a context carries about its task fixes the first limit where a count of the task's parameters cannot. Training longer and widening move the second, and show in the switch only where that limit is the tighter one.

4.3 Post-training moves a pretrained model to the side of its pool

A production LLM cannot be pretrained with a controlled pool, but it can be post-trained with one. We post-train Qwen2.5-0.5B and Qwen2.5-1.5B for 3000 steps on sequences half of natural text and half of four-sided-die episodes at $T = 128$ drawn from a pool of K tasks, and read the learning efficiency on new tasks before and after.

The pool decides which side the model ends on (Figure 7a). Four tasks pull both models down toward the exact memorizer of that pool, which takes 9.6%: from 82.9 to 49.2% and from 91.3 to 41.0%. Sixteen leave them where they began; sixty-four or more push them to 98.6–99.6%, above the memorizer of their own pool, so they end as learners of the family rather than stores of the pool. The turn falls between 16 and 64 tasks, where transformers trained from scratch on this family switch as well. The fall is selection, not damage (Figure 7b). Over the same post-training the held-out text loss improves in every pool, from 1.41 to about 1.2 nats per token, including the pool whose learning efficiency collapses, and the collapse is over within the first 500 steps and flat thereafter. And it acts one family at a time (Figure 7c). Post-training Qwen2.5-1.5B on two families at once, one from a pool of K tasks and one always from 4096, moves only the first: at $K = 4$ it falls from 92.3 and 90.9% to 50.1 and 57.1% while the other rises to about 99% in the same run,

and a third family never trained on is left where it was. We also find that the context length and the distribution over the pool, not only its size, decide where this turn falls: a longer context asks a pool for more entropy before memorizing stops paying (Figure 17), and skewing a pool of 4096 tasks selects memorizing without removing any of them (Figure 18).

The same selection acts on a model that is already trained, one family at a time.
Post-training a pretrained LLM on four tasks costs it half of what a new task's context offers, while its held-out text loss improves; sixty-four or more make it a near-ideal learner of the family instead. A family trained beside it from a large pool rises in the same run, and one never trained is untouched.

5 Related work

That pretraining switches from memorizing to learning as the task pool grows was established by Chan et al. (2022) and Raventós et al. (2023), extended by Kirsch et al. (2022), shown to follow how widely tasks are spread rather than how many there are (Goddard et al., 2025), counted for linear regression (Wu et al., 2024), and named as task retrieval against task learning (Lin & Lee, 2024). The switch has been explained dynamically (Nguyen & Reddy, 2025; Carroll et al., 2025), normatively with constants fitted to the runs (Wurgaft et al., 2025), and exactly in linear models, where it follows dimension or rank but not context length (Lu et al., 2025; Takanami et al., 2026; Kim et al., 2026); Gibson et al. (2026) separate a kinetic from a capacity threshold. We price those same two predictors in nats computed from the task family alone, so the switch is placed before training rather than fitted after it, and the capacity side appears as a second limit read off the trained model.

The identities we price with come from reading in-context learning as inference of a latent task (Xie et al., 2022; Wies et al., 2023; Jeon et al., 2024; Riechers et al., 2025; Wakayama & Suzuki, 2026; Arora et al., 2025; Elmoznino et al., 2025); we apply them to two predictors at once, which is what turns the threshold into a computable number. The engine itself is the information ratchet (Mandal & Jarzynski, 2012; Boyd et al., 2016; 2017b; 2022; Kolchinsky & Wolpert, 2017; Still et al., 2012; Boyd et al., 2025), from which we take the identity between work and prediction and the cost of a mismatched belief; none of it concerns neural networks, and the thermodynamic readings of language models that do (Sagawa, 2026; Halperin, 2025) extract no work and use no exact floor. Finite-memory estimation (Hellman & Cover, 1970; Leighton & Rivest, 1986; Berg et al., 2021) fixes what a bounded-state machine should reach, the gate of our memory experiment. Frozen models have been scored against exact Bayesian predictors (Falck et al., 2024; Chlon et al., 2025; Ozelim & Zenil, 2026; Sorensen et al., 2026); we add the split of what they lose into the part no model can remove and the part a better model could. Appendix A gives all of this at length.

6 Discussion

What the information ratchet supplies. Every identity used here has an information-theoretic proof and work is defined from the loss, so nothing in this paper tests a claim about heat. What the information ratchet supplies is a common currency: an ideal in-context learner, an ideal memorizer of a pool and a trained model are priced in the same nats, on the same sequences. That is what lets the switch be computed before a model exists, and what lets its two limits—what a context can distinguish and what a model can hold—be measured apart.

The bound of the research question we answered. Read in that currency, the learning toll is a floor: transformers trained on a fresh task in every sequence sit on it, frozen language models pay several times it, and what they overpay divides into a wrong starting belief and a failure to keep learning from the evidence. The same quantity counts tasks, so it also says where memorizing stops paying. Two things bound the account. (1) The exact toll exists only for families whose generating process is known, so applying it to models pretrained on natural text explains rather than tests. (2) What fixes how many tasks a model holds we cannot say, beyond that training length and width move it and that a fixed store per parameter does not account for it. In addition, whether this measurement also ranks how well a model uses a context in general is a separate question; the checks we can make are reported in Figure 19 and are negative.

AI USE STATEMENT

In this work, we used generative AI tools only for language polishing, improving the grammar, wording, and clarity of text drafted by the authors. We have not used generative AI tools for research ideation, for the design of the proposed method or the experimental protocol, for implementation, for the generation or analysis of results, for the creation of figures and tables, or for literature discovery and citation; all of these are the original work of the authors. We have reviewed all AI-assisted work, checking every suggested revision against the authors' original text and the underlying results, and we take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

Appendix C specifies every experiment: the task families and their priors, the architectures, optimizer, training lengths, seeds and evaluation splits, the protocol for scoring frozen models including the token mapping and the coverage gate, and the exact computation of the ideal memorizer and the ideal in-context learner, so that both references can be recomputed from a family definition without training anything. Complete proofs of Theorem 1, Proposition 1, Theorem 2 and the two lemmas they rest on are in Appendix B, together with the comparison of divergences and criteria that locate the switch. All frozen and post-trained models are public checkpoints, and all task families are synthetic and fully specified, so no dataset access is required. The criteria that locate the switch are stated in Appendix C; code for training, scoring and every figure will be released upon publication.

REFERENCES

Rishabh Agarwal, Avi Singh, Lei M Zhang, Bernd Bohnet, Luis Rosias, Stephanie CY Chan, et al. Many-shot in-context learning. In *Advances in Neural Information Processing Systems*, 2024.

Ekin Akyürek, Dale Schuurmans, Jacob Andreas, Tengyu Ma, and Denny Zhou. What learning algorithm is in-context learning? Investigations with linear models. In *International Conference on Learning Representations*, 2023.

Cem Anil, Esin Durmus, Nina Panickssery, et al. Many-shot jailbreaking. In *Advances in Neural Information Processing Systems*, 2024.

Aryaman Arora, Dan Jurafsky, Christopher Potts, and Noah D Goodman. Bayesian scaling laws for in-context learning. In *Conference on Language Modeling*, 2025.

Charles H Bennett. The thermodynamics of computation—a review. *International Journal of Theoretical Physics*, 1982.

Tomer Berg, Or Ordentlich, and Ofer Shayevitz. Deterministic finite-memory bias estimation. In *Conference on Learning Theory*, 2021.

William Bialek, Ilya Nemenman, and Naftali Tishby. Predictability, complexity, and learning. *Neural Computation*, 2001.

Alexander B Boyd, Dibyendu Mandal, and James P Crutchfield. Identifying functional thermodynamics in autonomous Maxwellian ratchets. *New Journal of Physics*, 2016.

Alexander B Boyd, Dibyendu Mandal, and James P Crutchfield. Correlation-powered information engines and the thermodynamics of self-correction. *Physical Review E*, 2017a.

Alexander B Boyd, Dibyendu Mandal, and James P Crutchfield. Leveraging environmental correlations: The thermodynamics of requisite variety. *Journal of Statistical Physics*, 2017b.

Alexander B Boyd, Dibyendu Mandal, and James P Crutchfield. Thermodynamics of modularity: Structural costs beyond the Landauer bound. *Physical Review X*, 2018.

Alexander B Boyd, James P Crutchfield, and Mile Gu. Thermodynamic machine learning through maximum work production. *New Journal of Physics*, 2022.

Alexander B Boyd, James P Crutchfield, Mile Gu, and Felix C Binder. Thermodynamic overfitting and generalization: energetics of predictive intelligence. *New Journal of Physics*, 2025.

Tom B. Brown et al. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, 2020.

Liam Carroll, Jesse Hoogland, Matthew Farrugia-Roberts, and Daniel Murfet. Dynamics of transient structure in in-context linear regression transformers. *arXiv preprint arXiv:2501.17745*, 2025.

Stephanie CY Chan, Adam Santoro, Andrew K Lampinen, Jane X Wang, Aaditya K Singh, Pierre H Richemond, James McClelland, and Felix Hill. Data distributional properties drive emergent in-context learning in transformers. In *Advances in Neural Information Processing Systems*, 2022.

Leon Chlon, Fatima Sheaib, Zein Khamis, Maggie Chlon, Mahdi El Zein, and MarcAntonio M. Awada. LLMs are Bayesian, in expectation, not in realization. *arXiv preprint arXiv:2507.11768*, 2025.

Bertrand S Clarke and Andrew R Barron. Information-theoretic asymptotics of Bayes methods. *IEEE Transactions on Information Theory*, 1990.

Thomas M Cover and Joy A Thomas. *Elements of information theory*. Wiley, 1991.

James P Crutchfield and David P Feldman. Regularities unseen, randomness observed: Levels of entropy convergence. *Chaos*, 2003.

James P Crutchfield and Karl Young. Inferring statistical complexity. *Physical Review Letters*, 1989.

Paul Cuff. Distributed channel synthesis. *IEEE Transactions on Information Theory*, 2013.

Eric Elmoznino, Tom Marty, Tejas Kasetty, Leo Gagnon, Sarthak Mittal, Mahan Fathi, Dhanya Sridhar, and Guillaume Lajoie. In-context learning and Occam's razor. In *International Conference on Machine Learning*, 2025.

Fabian Falck, Ziyu Wang, and Chris Holmes. Is in-context learning in large language models Bayesian? A martingale perspective. In *International Conference on Machine Learning*, 2024.

Shivam Garg, Dimitris Tsipras, Percy Liang, and Gregory Valiant. What can transformers learn in-context? A case study of simple function classes. In *Advances in Neural Information Processing Systems*, 2022.

Cole Gibson, Wenping Cui, and Gautam Reddy. Distinct mechanisms underlying in-context learning in transformers. *arXiv preprint arXiv:2604.12151*, 2026.

Chase Goddard, Lindsay M. Smith, Vudtiwat Ngampruetikorn, and David J. Schwab. When can in-context learning generalize out of task distribution? In *International Conference on Machine Learning*, 2025.

Igor Halperin. Semantic faithfulness and entropy production measures to tame your LLM demons and manage hallucinations. *arXiv preprint arXiv:2512.05156*, 2025.

Te Sun Han and Sergio Verdú. Approximation theory of output statistics. *IEEE Transactions on Information Theory*, 1993.

Martin E. Hellman and Thomas M. Cover. Learning with finite memory. *The Annals of Mathematical Statistics*, 1970.

Hong Jun Jeon, Jason D. Lee, Qi Lei, and Benjamin Van Roy. An information-theoretic analysis of in-context learning. In *International Conference on Machine Learning*, 2024.

John L Kelly. A new interpretation of information rate. *The Bell System Technical Journal*, 1956.

Daesik Kim, Sumin Choi, Hyojae Jeon, and Jung Hoon Han. Landau theory of quenched criticality in linear in-context learning. *arXiv preprint arXiv:2608.28059*, 2026.

Louis Kirsch, James Harrison, Jascha Sohl-Dickstein, and Luke Metz. General-purpose in-context learning by meta-learning transformers. *arXiv preprint arXiv:2212.04458*, 2022.

Artemy Kolchinsky and David H. Wolpert. Dependence of dissipation on the initial distribution over states. *Journal of Statistical Mechanics: Theory and Experiment*, 2017.

Fumiyasu Komaki. On asymptotic properties of predictive distributions. *Biometrika*, 1996.

Rolf Landauer. Irreversibility and heat generation in the computing process. *IBM Journal of Research and Development*, 1961.

Frank Thomson Leighton and Ronald L. Rivest. Estimating a probability using finite memory. *IEEE Transactions on Information Theory*, 1986.

Ziqian Lin and Kangwook Lee. Dual operating modes of in-context learning. In *International Conference on Machine Learning*, 2024.

Yue M Lu, Mary I Letey, Jacob A Zavatone-Veth, Anindita Maiti, and Cengiz Pehlevan. Asymptotic theory of in-context learning by linear attention. *Proceedings of the National Academy of Sciences*, 2025.

Dibyendu Mandal and Christopher Jarzynski. Work and information processing in a solvable model of Maxwell's demon. *Proceedings of the National Academy of Sciences*, 2012.

Gonzalo Manzano, Diego Subero, Olivier Maillet, Rosario Fazio, Jukka P Pekola, and Édgar Roldán. Thermodynamics of gambling demons. *Physical Review Letters*, 2021.

Neri Merhav. Sequence complexity and work extraction. *Journal of Statistical Mechanics: Theory and Experiment*, 2015.

Sewon Min, Xinxi Lyu, Ari Holtzman, Mikel Artetxe, Mike Lewis, Hannaneh Hajishirzi, and Luke Zettlemoyer. Rethinking the role of demonstrations: What makes in-context learning work? In *Conference on Empirical Methods in Natural Language Processing*, 2022.

Alex Nguyen and Gautam Reddy. Differential learning kinetics govern the transition from memorization to generalization during in-context learning. In *International Conference on Learning Representations*, 2025.

Luan Ozelim and Hector Zenil. Measuring in-context algorithmic reasoning in language models against an exact Bayes-optimal reference. *arXiv preprint arXiv:2608.01575*, 2026.

Core Francisco Park, Ekdeep Singh Lubana, Itamar Pres, and Hidenori Tanaka. Competition dynamics shape algorithmic phases of in-context learning. In *International Conference on Learning Representations*, 2025.

Juan MR Parrondo, Jordan M Horowitz, and Takahiro Sagawa. Thermodynamics of information. *Nature Physics*, 2015.

Mateusz Piotrowski, Paul M Riechers, Daniel Filan, and Adam S Shai. Constrained belief updates explain geometric structures in transformer representations. In *International Conference on Machine Learning*, 2025.

Allan Raventós, Mansheej Paul, Feng Chen, and Surya Ganguli. Pretraining task diversity and the emergence of non-Bayesian in-context learning for regression. In *Advances in Neural Information Processing Systems*, 2023.

Paul M Riechers, Henry R Bigelow, Eric A Alt, and Adam S Shai. Next-token pretraining implies in-context learning. *arXiv preprint arXiv:2505.18373*, 2025.

Takahiro Sagawa. Stochastic thermodynamics for autoregressive generative models: A non-Markovian perspective. *arXiv preprint arXiv:2604.07867*, 2026.

Takahiro Sagawa and Masahito Ueda. Generalized Jarzynski equality under nonequilibrium feedback control. *Physical Review Letters*, 2010.

Adam S Shai, Sarah E Marzen, Lucas Teixeira, Alexander G Oldenziel, and Paul M Riechers. Transformers represent belief state geometry in their residual stream. In *Advances in Neural Information Processing Systems*, 2024.

Aaditya K Singh, Stephanie CY Chan, Ted Moskovitz, Erin Grant, Andrew M Saxe, and Felix Hill. The transient nature of emergent in-context learning in transformers. In *Advances in Neural Information Processing Systems*, 2023.

Aaditya K Singh, Ted Moskovitz, Sara Dragutinović, Felix Hill, Stephanie CY Chan, and Andrew M Saxe. Strategy coopetition explains the emergence and transience of in-context learning. In *International Conference on Machine Learning*, 2025.

Taylor Sorensen, Benjamin Newman, Jared Moore, Chan Young Park, Jillian Fisher, Niloofar Mireshghallah, Liwei Jiang, and Yejin Choi. Spectrum tuning: Post-training for distributional coverage and in-context steerability. In *International Conference on Learning Representations*, 2026.

Susanne Still. Thermodynamic cost and benefit of memory. *Physical Review Letters*, 2020.

Susanne Still, David A Sivak, Anthony J Bell, and Gavin E Crooks. Thermodynamics of prediction. *Physical Review Letters*, 2012.

Leo Szilard. Über die Entropieverminderung in einem thermodynamischen System bei Eingriffen intelligenter Wesen. *Zeitschrift für Physik*, 1929.

Kaito Takanami, Takashi Takahashi, and Yoshiyuki Kabashima. Learning linear regression with low-rank tasks in-context. In *International Conference on Artificial Intelligence and Statistics*, 2026.

Léo Touzo, Matteo Marsili, Neri Merhav, and Édgar Roldán. Optimal work extraction and the minimum description length principle. *Journal of Statistical Mechanics: Theory and Experiment*, 2020.

Shoichi Toyabe, Takahiro Sagawa, Masahito Ueda, Eiro Muneyuki, and Masaki Sano. Experimental demonstration of information-to-energy conversion and validation of the generalized Jarzynski equality. *Nature Physics*, 2010.

Dror A Vinkler, Haim H Permuter, and Neri Merhav. Analogy between gambling and measurement-based work extraction. *Journal of Statistical Mechanics: Theory and Experiment*, 2016.

Johannes von Oswald, Eyvind Niklasson, Ettore Randazzo, João Sacramento, Alexander Mordvintsev, Andrey Zhmoginov, and Max Vladymyrov. Transformers learn in-context by gradient descent. In *International Conference on Machine Learning*, 2023.

Tomoya Wakayama and Taiji Suzuki. In-context learning is provably Bayesian inference: A generalization theory for meta-learning. In *International Conference on Machine Learning*, 2026.

Noam Wies, Yoav Levine, and Amnon Shashua. The learnability of in-context learning. In *Advances in Neural Information Processing Systems*, 2023.

David H Wolpert. The stochastic thermodynamics of computation. *Journal of Physics A: Mathematical and Theoretical*, 2019.

Jingfeng Wu, Difan Zou, Zixiang Chen, Vladimir Braverman, Quanquan Gu, and Peter L Bartlett. How many pretraining tasks are needed for in-context learning of linear regression? In *International Conference on Learning Representations*, 2024.

Daniel Wurgaft, Ekdeep Singh Lubana, Core Francisco Park, Hidenori Tanaka, Gautam Reddy, and Noah D Goodman. In-context learning strategies emerge rationally. In *Advances in Neural Information Processing Systems*, 2025.

Sang Michael Xie, Aditi Raghunathan, Percy Liang, and Tengyu Ma. An explanation of in-context learning as implicit Bayesian inference. In *International Conference on Learning Representations*, 2022.

Ruiqi Zhang, Spencer Frei, and Peter L Bartlett. Trained transformers learn linear models in-context. *Journal of Machine Learning Research*, 2024.

A FULL RELATED WORK

A.1 WHEN IN-CONTEXT LEARNING EMERGES

That pretraining switches from memorizing its task pool to learning each task from the context, as the pool grows, was established by Chan et al. (2022) and Raventós et al. (2023), who also identified the predictor below the switch as Bayesian inference over the pool, and extended by Kirsch et al. (2022) to a wider range of tasks and architectures. Goddard et al. (2025) show that what matters is how widely the tasks are spread rather than how many of them there are, and Wu et al. (2024) ask directly how many pretraining tasks in-context learning of linear regression requires. Lin & Lee (2024) name the two sides of the switch task retrieval and task learning and model them as a mixture prior, which is the same pair of predictors we price against each other. The switch need not be permanent: in-context learning can appear during training and then decay (Singh et al., 2023; 2025; Park et al., 2025). Longer contexts do not change the picture qualitatively but do keep paying: performance continues to improve with hundreds or thousands of demonstrations (Agarwal et al., 2024; Anil et al., 2024), which is the regime in which the toll's slow logarithmic growth matters most.

Explanations divide into three kinds. Dynamical accounts follow the two circuits through training and read the switch off their race (Nguyen & Reddy, 2025; Carroll et al., 2025). Normative accounts place it where the cheaper predictor changes hands, with constants fitted to the runs (Wurgaft et al., 2025). Exact accounts solve linear models, where the threshold follows the task dimension or the rank of the pool but not the length of the context (Lu et al., 2025; Takanami et al., 2026; Kim et al., 2026). Closest to the two limits of §4, Gibson et al. (2026) separate a kinetic threshold from a capacity threshold, both of which fall far below the exact crossing when a task carries much information.

What the ratchet reading adds is that the two predictors of this literature become two engines whose cost is measured in the same nats, computed from the task family before any model is trained. That places the switch without fitting anything to the runs, and it makes the growth of the switch with context length a property of the family and its prior rather than a single exponent: two families with almost the same number of task parameters grow at 2.76 and 1.45 nats per e-fold (§4.2). It also separates that limit from a second one, how many tasks a trained model holds, which the same measurement reads off the trained model itself.

A.2 HOW IN-CONTEXT LEARNING IS IMPLEMENTED

A parallel literature asks by what mechanism a transformer learns in context. Garg et al. (2022) showed that transformers trained on function classes match classical estimators on new functions; Akyürek et al. (2023) and von Oswald et al. (2023) argue that the forward pass implements an implicit learning algorithm, close to gradient descent or ridge regression on the demonstrations, and Zhang et al. (2024) prove convergence to such an algorithm for linear attention. Min et al. (2022) show that which parts of a demonstration matter is not what one would assume.

We make no claim of this kind. Everything here is read off predictions, so the same measurement applies to a model whose mechanism is unknown, including the frozen models of §3.2; what it reports is how far a model is from the best possible predictor, not which circuit puts it there. The two literatures meet at one point: a model that implements exact Bayesian updating on the demonstrations would have zero dissipation, so our dissipation is an upper bound on how well any such implicit algorithm can be doing.

A.3 BAYESIAN AND INFORMATION-THEORETIC ACCOUNTS

Reading in-context learning as inference of a latent task (Xie et al., 2022; Wies et al., 2023) gives the loss of an ideal learner in closed form. Jeon et al. (2024) decompose the Bayes-optimal loss into the loss of a predictor told the task and the information the context carries about it. The dichotomy between retrieving a stored task and learning a new one is drawn in information terms by Riechers et al. (2025) and Wakayama & Suzuki (2026). Bayesian scaling laws (Arora et al., 2025) and prequential coding (Elmoznino et al., 2025) supply the same identities in other notation. What we

add is the comparison: once both predictors are priced in nats per context, their difference is a number computable before training, and the threshold theorem of §4 follows from it.

A.4 Predictive information and computational mechanics

Bialek et al. (2001) isolate the predictive information of a sequence, the part of the entropy that is about the future rather than the past, and show that its growth classifies how learnable a source is; the toll $\Lambda(T) = I(\theta; X_{1:T})$ is the part of that information that concerns the task parameter, and its logarithmic growth in T is the case their analysis calls a finite-dimensional model class. Computational mechanics asks the memory question from the side of the process rather than the machine: how many states a predictor of a given source must keep, and how the entropy rate converges to its floor (Crutchfield & Young, 1989; Crutchfield & Feldman, 2003). Merhav (2015) connects sequence complexity to the work that can be extracted, which is the bridge between that literature and the engines below. What we take from this side is the confidence that the quantity we price in is the natural one, and a warning: the same information can be held in memories of very different size, which is why §4 measures what a model holds rather than inferring it.

A.5 Information engines and finite memory

Information thermodynamics—the accounting that makes Maxwell's demon consistent with the second law—is a settled field with standard reviews (Parrondo et al., 2015; Wolpert, 2019), an exact fluctuation theorem under feedback control (Sagawa & Ueda, 2010), an experimental realization of information-to-energy conversion (Toyabe et al., 2010), and a treatment of demons that bet rather than extract (Manzano et al., 2021), which is the form closest to the gambling identity of Kelly (1956) that §2 uses. Within it, the part we borrow is that of information ratchets: machines that convert the order of a tape into work (Mandal & Jarzynski, 2012; Boyd et al., 2016), the memory they need to do it (Boyd et al., 2017a;b; 2018), the equivalence between extracting the most work and predicting best (Boyd et al., 2022; Touzo et al., 2020), the cost of acting on a mismatched belief (Kolchinsky & Wolpert, 2017), thermodynamic overfitting (Boyd et al., 2025), and the cost of holding memory that does not predict (Still et al., 2012; Still, 2020). None of this work involves neural networks or in-context learning. Finite-memory estimation supplies the other side of the memory question: how well a machine with a bounded state can estimate a parameter from a stream (Hellman & Cover, 1970; Leighton & Rivest, 1986; Berg et al., 2021). These results fix what an exactly trained finite-state machine should reach. Thermodynamic language has been applied to language models before (Sagawa, 2026; Halperin, 2025). Those readings do not extract work, do not compare a model with an exact floor, and do not address in-context learning.

A.6 Frozen models against exact references

Pretrained language models have been scored against exact Bayesian predictors on synthetic tapes (Falck et al., 2024; Chlon et al., 2025; Ozelim & Zenil, 2026), and post-training is known to change how a model uses its context (Sorensen et al., 2026). We see the same deviations. What the learning toll adds is that the gap divides in two: the part no model can remove, which the task family fixes, and the part a better model could, which is the model's own. §3.2 measures both, and splits the second again into what a wrong starting belief costs and what remains.

B Proofs

B.1 Proof of Theorem 1

Throughout, X_t is the t-th example, and the LLM's prediction of it may depend on $X_{<t}$ but not on θ. Write

$$p_t(x) = \Pr(X_t = x \mid X_{<t}) = \mathbb{E}\big[P_\theta(x \mid X_{<t}) \,\big|\, X_{<t}\big], \qquad q_t(x) = q(x \mid X_{<t})$$

for the predictions of the ideal in-context learner (the posterior predictive) and of the LLM.

Step 1: the LLM's loss. Adding and subtracting $\ln p_t(X_t)$,

$$\ell_t = \mathbb{E}\big[-\ln q_t(X_t)\big] = \mathbb{E}\big[-\ln p_t(X_t)\big] + \mathbb{E}\Big[\ln \frac{p_t(X_t)}{q_t(X_t)}\Big].$$

Because p_t is the distribution of X_t given $X_{<t}$, the first term is $H(X_t \mid X_{<t})$. Conditioning the second term on $X_{<t}$, and using that q_t is fixed once $X_{<t}$ is,

$$\mathbb{E}\Big[\ln \frac{p_t(X_t)}{q_t(X_t)} \,\Big|\, X_{<t}\Big] = \sum_x p_t(x) \ln \frac{p_t(x)}{q_t(x)} = D_{\mathrm{KL}}(p_t \,\|\, q_t)\,.$$

Hence

$$\ell_t = H(X_t \mid X_{<t}) + \mathbb{E}\, D_{\mathrm{KL}}(p_t \,\|\, q_t)\,.$$

Step 2: identifying the dissipation. By the chain rule,

$$H(X_t \mid X_{<t}) = H(X_t \mid X_{<t}, \theta) + I(\theta; X_t \mid X_{<t}) = H(X_t \mid X_{<t}, \theta) + i_t,$$

so $\ell_t = H(X_t \mid X_{<t}, \theta) + i_t + \mathbb{E}\, D_{\mathrm{KL}}(p_t \,\|\, q_t)$, and comparison with equation 3 gives

$$d_t = \mathbb{E}\, D_{\mathrm{KL}}(p_t \,\|\, q_t)\,.$$

Step 3 uses the following standard inequality.

Lemma 1 (Information inequality; Cover & Thomas, 1991)**.** *For distributions p and q on the same space, $D_{\mathrm{KL}}(p \,\|\, q) \geq 0$, with equality if and only if $p = q$.*

Step 3: sign and equality. By Lemma 1, $D_{\mathrm{KL}}(p_t \,\|\, q_t) \geq 0$, with equality if and only if $q_t = p_t$. Therefore $d_t \geq 0$, with equality if and only if $q_t = p_t$ almost surely, that is, if the LLM predicts as the ideal in-context learner at position t.

Step 4: summing over the context. Summing equation 3 over $t = 1, \ldots, T$,

$$\sum_{t=1}^{T} \big(\ell_t - H(X_t \mid X_{<t}, \theta)\big) = \sum_{t=1}^{T} i_t + \sum_{t=1}^{T} d_t.$$

By the chain rule for mutual information,

$$\sum_{t=1}^{T} i_t = \sum_{t=1}^{T} I(\theta; X_t \mid X_{<t}) = I(\theta; X_{1:T}) = \Lambda(T),$$

which gives equation 5. Since every $d_t \geq 0$, the left side is at least $\Lambda(T)$, with equality if and only if $d_t = 0$ at every position. For real-valued examples, sums become integrals and entropies become differential entropies; every step is unchanged.

Step 5: work. For discrete examples over A values, equation 1 gives the expected work at position t as

$$W_t^{\mathrm{LLM}} = k_B T_{\mathrm{B}} \big(\ln A - \ell_t\big), \qquad W_t^{\mathrm{oracle}} = k_B T_{\mathrm{B}} \big(\ln A - H(X_t \mid X_{<t}, \theta)\big).$$

Summing the difference over the context and using equation 5,

$$\sum_{t=1}^{T} \big(W_t^{\mathrm{oracle}} - W_t^{\mathrm{LLM}}\big) = k_B T_{\mathrm{B}} \sum_{t=1}^{T} \big(\ell_t - H(X_t \mid X_{<t}, \theta)\big) \;\geq\; k_B T_{\mathrm{B}}\, \Lambda(T). \qquad \square$$

B.2 Proof of Proposition 1

Finite pool: the bound. For any process,

$$\Lambda(T) = I(\theta; X_{1:T}) = H(\theta) - H(\theta \mid X_{1:T}) \leq H(\theta) \leq \ln K,$$

because conditional entropy is non-negative and a variable with K values has entropy at most $\ln K$.

Finite pool: exponential decay. Let the examples be i.i.d. given the task, with distinct distributions $P_1, \ldots, P_K$ and prior weights $\pi_1, \ldots, \pi_K$. Since conditional entropy is non-negative,

$$i_t = I(\theta; X_t \mid X_{<t}) = H(\theta \mid X_{<t}) - H(\theta \mid X_{\leq t}) \leq H(\theta \mid X_{<t}).$$

Let $\hat{\theta}_n$ be the maximum a posteriori guess of the task from the first $n = t - 1$ examples, and let

$$\rho = \max_{j \neq k} \sum_x \sqrt{P_j(x)\, P_k(x)},$$

which is below 1 because the distributions are distinct. Given $\theta = j$, the guess can be wrong only if some $k \neq j$ has $\pi_k P_k^n(X_{<t}) \geq \pi_j P_j^n(X_{<t})$, where P^n is the distribution of n i.i.d. examples. By Markov's inequality applied to the square root of this ratio,

$$\Pr\left(\pi_k P_k^n \geq \pi_j P_j^n \,\middle|\, \theta = j\right) \leq \mathbb{E}_j\left[\sqrt{\frac{\pi_k P_k^n(X_{<t})}{\pi_j P_j^n(X_{<t})}}\right] = \sqrt{\frac{\pi_k}{\pi_j}}\Big(\sum_x \sqrt{P_j(x)P_k(x)}\Big)^n \leq \sqrt{\frac{\pi_k}{\pi_j}}\,\rho^n.$$

Summing over $k \neq j$, averaging over j, and using $\sqrt{\pi_j \pi_k} \leq (\pi_j + \pi_k)/2$,

$$e_n = \Pr(\hat{\theta}_n \neq \theta) \leq \sum_{j \neq k} \sqrt{\pi_j \pi_k}\,\rho^n \leq (K-1)\,\rho^n.$$

The remaining uncertainty about the task is controlled by the error probability through the following inequality.

Lemma 2 (Fano's inequality; Cover & Thomas, 1991)**.** *Let θ take K values and let $\hat{\theta}$ be any estimate of θ computed from an observation Y, with error probability $e = \Pr(\hat{\theta} \neq \theta)$. Then*

$$H(\theta \mid Y) \leq h(e) + e\ln(K-1), \qquad h(e) = -e\ln e - (1-e)\ln(1-e).$$

By Lemma 2 with $Y = X_{<t}$,

$$H(\theta \mid X_{<t}) \leq h(e_n) + e_n \ln(K-1).$$

Since $h(e) \leq e\ln(1/e) + e$ and $e\ln(1/e)$ increases for $e < 1/e$, once $(K-1)\rho^n < 1/e$,

$$i_t \leq H(\theta \mid X_{<t}) \leq (K-1)\,\rho^n\Big(n\ln\frac{1}{\rho} + 1\Big),$$

which decays exponentially in t at any rate below $\ln(1/\rho)$.

For a smooth family we use the following result.

Lemma 3 (Clarke & Barron, 1990)**.** *Let $X_1, X_2, \ldots$ be i.i.d. from P_θ, $\theta \in \Theta \subset \mathbb{R}^{\dim}$, where the family is identifiable, $D_{\mathrm{KL}}(P_\theta \,\|\, P_{\theta'})$ is twice continuously differentiable in θ' near θ, and the Fisher information $\mathcal{I}(\theta)$ is continuous and positive definite. Let π be a prior density that is continuous and positive, and let M_T be the marginal distribution of $X_{1:T}$ under π. Then, for θ in the interior of Θ,*

$$D_{\mathrm{KL}}\big(P_\theta^T \,\|\, M_T\big) = \frac{\dim}{2}\ln\frac{T}{2\pi e} + \frac{1}{2}\ln\det\mathcal{I}(\theta) + \ln\frac{1}{\pi(\theta)} + o(1),$$

uniformly over compact subsets of the interior of Θ.

Smooth family: the toll. Let the examples be i.i.d. given θ and let the conditions of Lemma 3 hold; as in that lemma, we take π to be supported on a compact subset of the interior of Θ, so that the convergence is uniform over the support. Since $I(\theta; X_{1:T})$ is the prior average of $D_{\mathrm{KL}}\big(P_\theta^T \,\|\, M_T\big)$, averaging Lemma 3 over π gives

$$\Lambda(T) = I(\theta; X_{1:T}) = \mathbb{E}_\pi\, D_{\mathrm{KL}}\big(P_\theta^T \,\|\, M_T\big) = \frac{\dim}{2}\ln T + c_\pi + o(1),$$

with

$$c_\pi = \frac{\dim}{2}\ln\frac{1}{2\pi e} + \mathbb{E}_\pi\Big[\ln\frac{\sqrt{\det\mathcal{I}(\theta)}}{\pi(\theta)}\Big].$$

The Dirichlet prior of our die tasks is supported on a compact simplex, but it puts mass arbitrarily close to the boundary, where the Fisher information diverges, so it is not supported on a compact subset of the interior; and the Gaussian prior of our regression tasks is not compactly supported at all. For both, the same form with this constant agrees with the exact toll (Appendix D.2).

For the toll per example we use the following result.

Lemma 4 (Komaki, 1996)**.** *Under the conditions of Lemma 3, let $p_n(x) = \int P_\theta(x)\,\pi(\theta \mid X_{1:n})\,d\theta$ be the Bayes predictive density after n observations. Then, for θ in the interior of Θ,*

$$\mathbb{E}_\theta\, D_{\mathrm{KL}}(P_\theta \,\|\, p_n) = \frac{\dim}{2n} + o\Big(\frac{1}{n}\Big).$$

Smooth family: the toll per example. Because examples are i.i.d. given the task, $H(X_t \mid X_{<t}, \theta) = H(X_t \mid \theta)$, and therefore

$$i_t = H(X_t \mid X_{<t}) - H(X_t \mid \theta) = \mathbb{E}\Big[\ln \frac{P_\theta(X_t)}{p_t(X_t)}\Big] = \mathbb{E}\, D_{\mathrm{KL}}(P_\theta \,\|\, p_t)\,,$$

the Kullback–Leibler risk of the Bayes predictive density p_t after $n = t-1$ examples. By Lemma 4, averaged over π with the uniformity above,

$$\mathbb{E}\, D_{\mathrm{KL}}(P_\theta \,\|\, p_t) = \frac{\dim}{2n} + o\Big(\frac{1}{n}\Big).$$

The leading term has a direct reading. To second order, $D_{\mathrm{KL}}(P_\theta \,\|\, P_{\theta'}) \approx \frac{1}{2}(\theta' - \theta)^\top \mathcal{I}(\theta)(\theta' - \theta)$, and the posterior concentrates around θ with covariance $\mathcal{I}(\theta)^{-1}/n$, so the expected divergence is $\frac{1}{2}\operatorname{tr}\big(\mathcal{I}(\theta)\,\mathcal{I}(\theta)^{-1}/n\big) = \dim/2n$. Since $\frac{\dim}{2(t-1)} = \frac{\dim}{2t}\,(1 + o(1))$,

$$i_t = \frac{\dim}{2t}\,\big(1 + o(1)\big),$$

and summing over $t \le T$ recovers the leading term $\frac{\dim}{2}\ln T$ of $\Lambda(T)$. □

B.3 Setting and notation for Section 4

Examples take values in a finite set, so a context $x = x_{1:T}$ ranges over a finite set $\mathcal{X}^T$. For a task θ, P_θ^T is the distribution of $X_{1:T}$. A pool $\mathcal{K} = (\theta_1, \ldots, \theta_K)$ is drawn i.i.d. from π; a pretraining episode picks an index κ uniformly from $\{1, \ldots, K\}$ and draws $X_{1:T} \sim P_{\theta_\kappa}^T$. Write

$$P_{\mathcal{K}}(x) = \frac{1}{K}\sum_{k=1}^{K} P_{\theta_k}^T(x), \qquad M_\pi(x) = \mathbb{E}_{\theta\sim\pi}\big[P_\theta^T(x)\big]$$

for the distribution of a pretraining context given the pool and for the marginal distribution under the continuous prior. By the chain rule of probability, a predictor whose predictions multiply to $Q(x)$ has cumulative log-loss $-\ln Q(x)$ over the context. The oracle model predicts with $P_{\theta_\kappa}^T$, the memorizing demon $\mathcal{D}_K$ with $P_{\mathcal{K}}$, and the learning demon $\mathcal{D}_\pi$ with M_π. The learning tolls are

$$\Lambda_K(T) = \mathbb{E}\Big[\ln \frac{P_{\theta_\kappa}^T(X)}{P_{\mathcal{K}}(X)} \,\Big|\, \mathcal{K}\Big] = I(\kappa; X_{1:T} \mid \mathcal{K}),$$
$$\Lambda_\pi(T) = \mathbb{E}_{\theta\sim\pi}\, D_{\mathrm{KL}}\big(P_\theta^T \,\|\, M_\pi\big) = I(\theta; X_{1:T}).$$

B.4 Proof of Lemma 5

Lemma 5 (Retrieval margin)**.** $\Delta(K,T) := \Lambda_\pi(T) - \mathbb{E}_{\{\theta_k\}}[\Lambda_K(T)] \ge 0$*: on its own training sequences the ideal memorizer loses exactly* Δ *nats less than the ideal in-context learner.*

Step 1: the margin given the pool. On a pretraining episode, the cumulative loss of $\mathcal{D}_\pi$ minus that of $\mathcal{D}_K$ is $\ln P_{\mathcal{K}}(X) - \ln M_\pi(X)$. Given $\mathcal{K}$, $X \sim P_{\mathcal{K}}$, so its expectation is

$$\mathbb{E}\big[\ln P_{\mathcal{K}}(X) - \ln M_\pi(X) \,\big|\, \mathcal{K}\big] = \sum_x P_{\mathcal{K}}(x) \ln \frac{P_{\mathcal{K}}(x)}{M_\pi(x)} = D_{\mathrm{KL}}(P_{\mathcal{K}} \,\|\, M_\pi) \ge 0,$$

by Lemma 1.

Step 2: both excesses over the oracle model. Given $\mathcal{K}$, the excess of $\mathcal{D}_K$ over the oracle model is $\Lambda_K(T)$ by definition. The excess of $\mathcal{D}_\pi$ is $\mathbb{E}[\ln P_{\theta_\kappa}^T(X) - \ln M_\pi(X) \mid \mathcal{K}]$. Averaged over the pool, the task of a pretraining episode, θ_κ, is distributed as π, because the atoms are i.i.d. from π and κ is independent of them. Hence

$$\mathbb{E}_{\mathcal{K}}\,\mathbb{E}\big[\ln P_{\theta_\kappa}^T(X) - \ln M_\pi(X) \,\big|\, \mathcal{K}\big] = \mathbb{E}_{\theta\sim\pi}\, D_{\mathrm{KL}}\big(P_\theta^T \,\|\, M_\pi\big) = \Lambda_\pi(T).$$

Step 3: the margin. Subtracting the two excesses and averaging over the pool,

$$\Delta(K,T) = \Lambda_\pi(T) - \mathbb{E}_{\mathcal{K}}\big[\Lambda_K(T)\big] = \mathbb{E}_{\mathcal{K}}\, D_{\mathrm{KL}}(P_{\mathcal{K}} \,\|\, M_\pi) \ge 0,$$

which is the expected number of nats by which $\mathcal{D}_K$ beats $\mathcal{D}_\pi$ on a pretraining episode. □

B.5 Proof of Lemma 6

Lemma 6 (Resolvability). $\mathbb{E}[\Lambda_K(T)] \le \min(\ln K, \Lambda_\pi(T))$, *and* $\Delta(K,T) \to 0$ *as* $K \to \infty$ *at fixed* T.

Step 1: the bound $\ln K$. Given the pool, $\Lambda_K(T) = I(\kappa; X_{1:T} \mid \mathcal{K})$ is the toll of a task that takes K values, so by Proposition 1

$$\Lambda_K(T) \le H(\kappa \mid \mathcal{K}) = \ln K.$$

Together with Lemma 5, $\mathbb{E}[\Lambda_K(T)] \le \min\big(\ln K, \Lambda_\pi(T)\big)$.

Step 2: a divergence bound. For distributions P and M on a finite set, Jensen's inequality applied to the concave logarithm gives

$$D_{\mathrm{KL}}(P \,\|\, M) = \sum_x P(x) \ln \frac{P(x)}{M(x)} \le \ln \sum_x \frac{P(x)^2}{M(x)}.$$

Step 3: averaging over the pool. Applying Step 2 to $P_{\mathcal{K}}$ and M_π and Jensen's inequality again, now over $\mathcal{K}$,

$$\Delta(K,T) = \mathbb{E}_{\mathcal{K}}\, D_{\mathrm{KL}}(P_{\mathcal{K}} \,\|\, M_\pi) \le \ln \sum_x \frac{\mathbb{E}_{\mathcal{K}}\big[P_{\mathcal{K}}(x)^2\big]}{M_\pi(x)}.$$

Because $P_{\mathcal{K}}(x)$ is the mean of K i.i.d. draws of $P_\theta^T(x)$, whose mean is $M_\pi(x)$,

$$\mathbb{E}_{\mathcal{K}}\big[P_{\mathcal{K}}(x)^2\big] = M_\pi(x)^2 + \frac{1}{K}\Big(\mathbb{E}_\theta\big[P_\theta^T(x)^2\big] - M_\pi(x)^2\Big).$$

Summing over x and writing

$$e^{J_2(T)} = \sum_x \frac{\mathbb{E}_\theta\big[P_\theta^T(x)^2\big]}{M_\pi(x)},$$

which is finite because $\mathcal{X}^T$ is finite and $M_\pi(x) > 0$, gives

$$\Delta(K,T) \le \ln\Big(1 + \frac{e^{J_2(T)} - 1}{K}\Big) \le \ln\big(1 + e^{J_2(T) - \ln K}\big). \tag{7}$$

The right side tends to 0 as $K \to \infty$ at fixed T. □

Remark. $\Delta = \mathbb{E}_{\mathcal{K}}\, D_{\mathrm{KL}}(P_{\mathcal{K}} \,\|\, M_\pi)$ is the channel-resolvability quantity of Han & Verdú (1993), and equation 7 is an order-2 soft-covering bound (Cuff, 2013). A heuristic rate, not used below: for a $\dim$-dimensional family the nearest atom to a typical task lies at distance $O(K^{-1/\dim})$, and the per-example divergence between their predictions is $O(K^{-2/\dim})$, so $\Delta(K,T) = O(TK^{-2/\dim}) \wedge \Lambda_\pi(T)$.

B.6 Rényi redundancy of the Dirichlet–categorical family

Lemma 7 (Rényi redundancy of the Dirichlet–categorical family). *Let* $\theta \sim \mathrm{Dir}(\mathbf{1})$ *on* Δ^{A-1} *and let* $X_{1:T}$ *be i.i.d.* $\mathrm{Cat}(\theta)$ *given* θ. *Then* $J_2(T)$ *of equation 7 is, exactly,*

$$e^{J_2(T)} = \frac{\Gamma(T+A)\, 4^T\, \Gamma(T + A/2)}{\Gamma(2T + A)\, \Gamma(A/2)},$$

and, as $T \to \infty$,

$$J_2(T) = \frac{A-1}{2} \ln T + c_2(A) + o(1), \qquad c_2(A) = \tfrac{1}{2} \ln 2\pi - \big(A - \tfrac{1}{2}\big) \ln 2 - \ln \Gamma\big(\tfrac{A}{2}\big).$$

Proof. Step 1: terms of a sequence with given counts. Let x have counts $n = (n_1, \dots, n_A)$, with $\sum_a n_a = T$. Integrating against the $\mathrm{Dir}(\mathbf{1})$ density $\Gamma(A)$,

$$M_\pi(x) = \Gamma(A) \int \prod_a \theta_a^{n_a}\, d\theta = \frac{\Gamma(A) \prod_a n_a!}{\Gamma(T + A)},$$

$$\mathbb{E}_\theta\big[P_\theta^T(x)^2\big] = \Gamma(A) \int \prod_a \theta_a^{2n_a}\, d\theta = \frac{\Gamma(A) \prod_a (2n_a)!}{\Gamma(2T + A)}.$$

Their ratio is

$$\frac{\mathbb{E}_\theta\big[P_\theta^T(x)^2\big]}{M_\pi(x)} = \frac{\Gamma(T+A)}{\Gamma(2T+A)}\prod_a \frac{(2n_a)!}{n_a!}.$$

Step 2: summing over sequences. There are $T!/\prod_a n_a!$ sequences with counts n, so

$$e^{J_2(T)} = \sum_{|n|=T} \frac{T!}{\prod_a n_a!}\,\frac{\Gamma(T+A)}{\Gamma(2T+A)}\prod_a \frac{(2n_a)!}{n_a!} = \frac{T!\,\Gamma(T+A)}{\Gamma(2T+A)}\sum_{|n|=T}\prod_a \binom{2n_a}{n_a}.$$

Step 3: the sum over counts. Since $\sum_{k\ge 0}\binom{2k}{k}z^k = (1-4z)^{-1/2}$, the sum over counts is the coefficient of z^T in the A-th power of this series:

$$\sum_{|n|=T}\prod_a \binom{2n_a}{n_a} = [z^T]\,(1-4z)^{-A/2} = 4^T\,\frac{\Gamma(T+A/2)}{\Gamma(A/2)\,T!}.$$

Substituting into Step 2 gives the exact formula.

Step 4: the asymptotic form. By Stirling's formula, $\ln\Gamma(z+a) = (z+a-\frac{1}{2})\ln z - z + \frac{1}{2}\ln 2\pi + o(1)$ as $z\to\infty$. Applying it to each Gamma function, with $\ln\Gamma(2T+A) = (2T+A-\frac{1}{2})(\ln T+\ln 2) - 2T + \frac{1}{2}\ln 2\pi + o(1)$,

$$\begin{aligned} J_2(T) &= \Big[\big(T+A-\tfrac{1}{2}\big) + \big(T+\tfrac{A}{2}-\tfrac{1}{2}\big) - \big(2T+A-\tfrac{1}{2}\big)\Big]\ln T + 2T\ln 2 - \big(2T+A-\tfrac{1}{2}\big)\ln 2 \\ &\quad + \tfrac{1}{2}\ln 2\pi - \ln\Gamma\big(\tfrac{A}{2}\big) + o(1) = \frac{A-1}{2}\ln T + c_2(A) + o(1). \end{aligned}$$

□

We checked the exact formula against brute-force enumeration (agreement to 10^{-6} at small T). Since $\Lambda_\pi(T) = \frac{\dim}{2}\ln T + c_\pi + o(1)$ with $\dim = A-1$ (Proposition 1), J_2 and Λ_π share their leading term and differ by $c_2(A) - c_\pi(A)$ in the limit, 0.30 nats for $A=2$ and 0.87 for $A=4$; at $\dim = 3$ the exact difference is 0.74, 0.83 and 0.86 at $T = 32$, 128 and 512.

B.7 Proof of Theorem 2

Step 1: the switch is a single crossing. The margin does not increase with K. A pool of $K+1$ atoms averages the $K+1$ pools of K atoms obtained by leaving one atom out,

$$P_{\mathcal{K}_{K+1}} = \frac{1}{K+1}\sum_{j=1}^{K+1} P_{\mathcal{K}_{K+1}\setminus\{\theta_j\}},$$

and each leave-one-out pool is distributed as a pool of K atoms. We use the following property of the divergence.

Lemma 8 (Convexity of the divergence; Cover & Thomas, 1991)**.** *For a fixed distribution q, $D_{\mathrm{KL}}(p\,\|\,q)$ is convex in p.*

By Lemma 8,

$$\Delta(K+1,T) = \mathbb{E}\,D_{\mathrm{KL}}\big(P_{\mathcal{K}_{K+1}}\,\|\,M_\pi\big) \le \frac{1}{K+1}\sum_{j=1}^{K+1}\mathbb{E}\,D_{\mathrm{KL}}\big(P_{\mathcal{K}_{K+1}\setminus\{\theta_j\}}\,\|\,M_\pi\big) = \Delta(K,T).$$

Under the rule "memorize if and only if $\Delta(K,T) > c_{\mathrm{pen}}$", pretraining therefore memorizes for all K below

$$K^*(T) = \min\big\{K : \Delta(K,T) \le c_{\mathrm{pen}}\big\}$$

and learns from K^* on: the switch is where the exact margin falls through c_{pen}.

Step 2: a lower bound on the switch. By Lemmas 5 and 6,

$$\Delta(K,T) = \Lambda_\pi(T) - \mathbb{E}\big[\Lambda_K(T)\big] \ge \Lambda_\pi(T) - \ln K.$$

If $\ln K < \Lambda_\pi(T) - c_{\mathrm{pen}}$, then $\Delta(K,T) > c_{\mathrm{pen}}$ and $K < K^*$. Hence

$$\ln K^*(T) \geq \Lambda_\pi(T) - c_{\mathrm{pen}}.$$

Step 3: an upper bound on the switch. By equation 7, $\Delta(K,T) \leq c_{\mathrm{pen}}$ as soon as $e^{J_2(T)-\ln K} \leq e^{c_{\mathrm{pen}}} - 1$. The smallest such K is at most $\lceil e^{J_2(T)}/(e^{c_{\mathrm{pen}}}-1) \rceil$, so, when this is at least 2,

$$\ln K^*(T) \leq J_2(T) + \ln \frac{1}{e^{c_{\mathrm{pen}}}-1} + \ln 2.$$

Step 4: Dirichlet–categorical families. By Lemma 7 and Proposition 1, $J_2(T) - \Lambda_\pi(T) = c_2(A) - c_\pi(A) + o(1)$. Steps 2 and 3 then give

$$-c_{\mathrm{pen}} \;\leq\; \ln K^*(T) - \Lambda_\pi(T) \;\leq\; c_2(A) - c_\pi(A) + \ln \frac{1}{e^{c_{\mathrm{pen}}}-1} + \ln 2 + o(1),$$

so $\ln K^* = \Lambda_\pi(T) + O(1)$, with both constants explicit.

Step 5: training that does not reach the memorizing solution. If a training protocol fails to reach $\mathcal{D}_K$ at some $K < K^*$, the model it produces is classified as learning at that K, so the switch it exhibits can only move to smaller K. □

B.8 The $O(1)$ constant, and which criterion locates the switch

The theorem is stated for the margin $\Delta(K,T) = \mathbb{E}_{\mathcal{K}} D_{\mathrm{KL}}(P_{\mathcal{K}} \,\|\, M_\pi)$, which is evaluated on sequences whose task comes from the pool, whereas §4 locates the switch by the memorizer's dissipation on new tasks, $\mathbb{E}_{\mathcal{K}} D_{\mathrm{KL}}(M_\pi \,\|\, P_{\mathcal{K}})$, the quantity of §3. These are different divergences, and the level at which the crossing is read off, one nat per context, is a choice. We computed both on the same pools for Dirichlet–categorical families at three levels (2048 episodes, 4 pool draws, atoms nested in K; the margin curve at $K = 1$ returns $\Lambda_\pi(T)$ to within 0.05 nats, which checks the estimator).

Both choices move the offset and neither moves the slope. Regressing $\ln K^*$ on $\Lambda_\pi(T)$ within a family gives a slope between 0.98 and 1.07 in all twelve combinations of family, divergence and criterion, while the offset shifts by about a nat when the criterion is halved or doubled, and by 0.22 nats at $\dim = 7$ or 0.83 nats at $\dim = 3$ when the margin replaces the dissipation. Within a criterion the offset does not drift with context length: over the lengths in Table 1 its standard deviation is 0.05–0.16 nats. This is the $O(1)$ constant of the theorem, measured.

The level is a choice; its unit is not. A level of the memorizer's learning efficiency on a new task, $\eta_T = 1 - d_T/g_T$ (§4), reads the switch per symbol at the end of the context instead of per context, and read that way the exact switch barely grows with the context. On the pools of §4, and on a two-state hidden-Markov sweep run with the same protocol, the slope of $\ln K^*$ on $\Lambda_\pi(T)$ within a family is 0.06, 0.18 and 0.34 at $\dim = 3$ for efficiencies of 0.5, 0.9 and 0.99, 0.13, 0.32 and 0.53 at $\dim = 7$, and 0.03, 0.26 and 0.55 for the two-state hidden-Markov family, against 1.04, 1.04 and 1.10 for the margin at one nat. Late in a long context the memorizer's handicap per symbol is set by how far a new task lies from the nearest stored one, which depends on K and not on T. Storing the pool pays once per context, and c_{pen} is counted in the same unit. A criterion on a fraction of the toll fails for a related reason: requiring $\Delta(K,T) \leq \tau\, \Lambda_\pi(T)$ places the switch at $\ln K = (1-\tau)\Lambda_\pi(T)$ under the binding branch of Lemma 6, a slope below one by construction.

C Experimental setting

C.1 The task families

A task family fixes a prior π over tasks and, for each task, a distribution over sequences of demonstrations. A demonstration is one position of a context: a symbol in the discrete families, an input–output pair in regression. Table 2 lists the eight families used in the paper with the number of free parameters $\dim$ of a task in each.

In the dice families a task is a categorical distribution θ on A symbols drawn from $\mathrm{Dir}(\mathbf{1})$, and the symbols of a context are i.i.d. $\mathrm{Cat}(\theta)$, so $\dim = A - 1$. In the Markov family a task is a

Table 1: **The switch under two divergences and three criteria.** Entries are $\ln K^* - \Lambda_\pi(T)$ in nats, where K^* is the pool size at which the dissipation on new tasks (diss.) or the margin on pretraining sequences (marg.) first falls below c nats per context.

			$c = 0.5$		$c = 1$		$c = 2$	
family	T	$\Lambda_\pi(T)$	diss.	marg.	diss.	marg.	diss.	marg.
Dice - 4 sides (dim = 3)	32	3.03	+0.88	+0.38	+0.19	−0.52	−0.60	−1.88
	128	4.94	+1.01	+0.51	+0.39	−0.48	−0.34	−1.79
	512	6.97	+1.11	+0.50	+0.41	−0.51	−0.31	−1.85
Dice - 8 sides (dim = 7)	16	3.50	+0.72	+0.68	−0.22	−0.44	−1.11	−1.83
	32	5.13	+0.93	+0.84	−0.10	−0.29	−1.17	−1.74
	64	7.06	+1.00	+0.87	−0.02	−0.26	−1.12	−1.73
	128	9.30	+0.93	+0.87	−0.06	−0.28	−1.22	−1.77

transition matrix whose rows are drawn independently from $\mathrm{Dir}(\mathbf{1})$, the first symbol is uniform, and $\dim = A(A-1)$. In the hidden-Markov families a task is a pair of matrices, $S \times S$ transitions and $S \times A$ emissions, every row drawn from $\mathrm{Dir}(\mathbf{1})$, with a uniform distribution over the first hidden state; the free parameters are $S(S-1)$ in the transitions and $S(A-1)$ in the emissions. In the regression families a task is a weight vector $w \sim \mathcal{N}(0, I_{\dim})$, a demonstration is a pair (x, y) with $x \sim \mathcal{N}(0, I_{\dim})$ and $y = w^\top x + \varepsilon$, $\varepsilon \sim \mathcal{N}(0, \sigma^2)$ and $\sigma^2 = 1/4$. These families are not models of natural text: a sequence carries a single task of a few parameters, its symbols are exchangeable or Markov given that task, and the alphabet is a handful of symbols rather than a vocabulary. What they buy is the one property natural text lacks—the generating process is known, so the oracle model and the ideal in-context learner can be computed exactly instead of estimated, and every number in this paper is a distance to one of them.

Example. The four kinds of family differ in what a context reveals and in what has to be inferred from it. A task drawn from the four-sided die may be $\theta = (0.03, 0.10, 0.34, 0.54)$, and a context from it reads

B C D C D B C D C D D D C D D C A D C C

where all that can be learned is how often each symbol occurs, and the ideal learner's next prediction is the add-one frequency of the symbols so far. A task from the Markov family is a transition matrix, for instance with rows $(0.22, 0.24, 0.54)$, $(0.63, 0.03, 0.35)$ and $(0.33, 0.21, 0.46)$, whose contexts read

B A B C B C C C A C B C B A A B A A C B C C C C

and here the symbol frequencies are not enough: what must be learned is which symbol follows which, so the toll is larger at the same alphabet size. A task from the hidden-Markov family is a pair of matrices, and its contexts look no different—

A D C A C D C C A C C A A D C D C C D A A D C A

—but the state that generated each symbol is never shown, so the same symbol can come from either state and neither matrix can be read off the counts; this is the only family for which the ideal learner has no closed form. A regression task is a weight vector, for instance $w = (-0.80, -1.32)$, and a context is a list of pairs, $x = (-0.25, 0.42)$ with $y = -0.22$, then $x = (1.14, 0.11)$ with $y = -1.67$, and so on; here nothing can be inferred until $\dim$ pairs have been seen, which is where dissipation peaks.

In every family the model sees only the context, never the task, and a fresh task is drawn for every sequence unless a pool is fixed.

C.2 The reference predictors

Every measurement in this paper is a distance, in nats, between a trained model and one of three predictors computed on the same sequences. All three are exact, in the sense that they are the

Table 2: **The eight task families.** $\dim$ is the number of free parameters of one task, which fixes how fast the learning toll grows with context length (Proposition 1).

family	a task is	alphabet	$\dim$
Dice - 4 sides	a categorical distribution	$A = 4$	3
Dice - 8 sides	a categorical distribution	$A = 8$	7
Dice - 16 sides	a categorical distribution	$A = 16$	15
Markov chain	a transition matrix	$A = 3$	6
Hidden Markov - 2 states	transitions and emissions	$A = 4$	8
Hidden Markov - 4 states	transitions and emissions	$A = 4$	24
Regression - 2 dims	a weight vector in $\mathbb{R}^2$	—	2
Regression - 4 dims	a weight vector in $\mathbb{R}^4$	—	4

optimal predictors under what they are given, and they are evaluated position by position on the very episodes a model is scored on.

The oracle model. The oracle model is told the task and predicts with P_θ: the categorical distribution itself in the dice families, the row of the transition matrix selected by the previous symbol in the Markov family, the forward recursion over hidden states given the true transition and emission matrices in the hidden-Markov families, and $\mathcal{N}(w^\top x, \sigma^2)$ in regression. Knowing the task does not make its loss zero—it still pays the noise in the process—and that loss is the floor from which everything else in the ledger of equation 3 is measured.

The ideal in-context learner. The ideal in-context learner is told only the family and predicts with the posterior predictive under π, so it pays the learning toll on top of the oracle model and nothing more. For the dice and the Markov chain it is available in closed form: the $\mathrm{Dir}(\alpha)$ posterior predictive is add-α smoothing of the running symbol or transition counts, with $\alpha = 1$ except in the two sparser priors of §4.2. For regression it is ridge regression with the prior as regularizer, whose predictive variance is $\sigma^2 + x^\top \Sigma x$ with Σ the posterior covariance. For the hidden-Markov families no closed form exists, and we use the Bayes mixture over a reference library of $32{,}768$ tasks drawn from π; that library resolves the cumulative toll to within 0.02 nats up to $T = 64$ and slightly overestimates it at longer lengths (Appendix D.2), which is the one place in this paper where a reference is approximated rather than computed.

The ideal memorizer. The ideal memorizer of a pool of K tasks holds the pool in place of the family: it carries the K tasks themselves and predicts with the exact posterior over its members, updated multiplicatively per symbol and computed in float64, so that a context only has to say which member it faces. It is defined only where a pool is defined, that is in §4, and it is given the pool the model beside it was trained on. Both learning tolls are Monte-Carlo estimates over $2{,}048$ episodes of quantities that are exact per episode; their standard errors are below the line widths of every figure. A trained model is placed between this predictor and the ideal in-context learner by $\varphi = d_{\mathrm{mem}}/\Delta$ (§4), and the pool at which φ passes $\frac{1}{2}$ is the number of tasks the model holds.

C.3 Architecture and protocols

All trained models are GPT-style decoders: token and learned positional embeddings, pre-LayerNorm blocks, MLP ratio 4, causal masking, and a loss on every position, the first predicted from a zero "BOS" embedding, three seeds. They are trained with AdamW at learning rate 3×10^{-4}, linear warmup over 500 steps, no weight decay, gradient clipping at 1.0, float32, and a batch of 64 episodes sampled fresh at every step; when a pool is fixed, the task of an episode is drawn uniformly from it. The sweep of §3.1 trains one model per family and context length, 4 layers, $d_{\mathrm{model}} = 128$, 4 heads, 100k steps, a fresh task in every episode. The switch sweeps of §4.1 use the same architecture for 25k steps, over pools of $K = 2$ to $131{,}072$ tasks. The model-side runs of §4.2 repeat that protocol for 100k steps, and separately at $d_{\mathrm{model}} = 64$, 256 and 512. Evaluation always uses fixed held-out sets of $1{,}024$ episodes, split into episodes drawn from the pretraining pool and episodes of fresh tasks from π. Every training run in this paper used a single NVIDIA A40 or A100 GPU.

C.4 READING THE MEASUREMENTS

Dissipation is measured at each position as the divergence between two full predictive distributions, that of the ideal in-context learner and that of the model, averaged over the held-out sequences; it is not a difference of measured losses. The two estimators have the same expectation, but this one is non-negative at every sequence and position and has far lower variance. For regression, which has no discrete alphabet, the squared deviation of the model from the ridge prediction is divided by twice the ideal learner's predictive variance, which puts dissipation and toll on one log-loss ledger, so that the excess over the oracle model is again their sum. Learning efficiencies quoted at the end of a context average the gain and the dissipation over its last eighth. Curves along a context are averaged over log-spaced bins of demonstrations. In Figure 3b the efficiency is undefined at the first demonstration, where nothing has yet been provided; it is drawn from the second demonstration on the dice and regression, whose tolls are exact, and from the eighth on the Markov and hidden-Markov families, whose tolls are estimated on the evaluation sequences and are too noisy earlier.

C.5 PRETRAINED LANGUAGE MODELS

Scoring frozen models. Frozen models are evaluated with `transformers` on identical episodes from three processes (A=4 and A=8 i.i.d., A=3 Markov), 256 episodes per process. Symbols are mapped to verified single tokens (primary: " A"–" H"; control: rarer letters " Q, X, Z, J, V, W, P, M"), and sequences are prefixed with the tokenizer's BOS or a newline. Next-token distributions are restricted to the A symbol tokens and renormalized, which is meaningful only if a model puts most of its mass on those symbols, so a model is scored on a task only if its coverage there is at least 0.90; the gate matters, since coverage correlates with measured dissipation ($\rho = -0.53$). The unconstrained mass on the symbol set is 94–96% for Qwen and Pythia-1.4B and 91% for GPT-2 at A=8, but only 56–89% for Pythia-410M, whose numbers carry that caveat. The effective prior of a model is the add-α rule closest to it: a grid search over $\alpha \in [10^{-2}, 10^{3}]$ minimizing the squared distance between the two loss curves over positions $t \geq 3$.

Post-training. Qwen2.5-0.5B and Qwen2.5-1.5B are post-trained with AdamW at learning rate 10^{-5} with no schedule, batch 8, on sequences that are half natural text and half episodes of the four-sided die at $T = 128$ drawn from a pool of K tasks, for 3,000 steps. Weights are bf16, with four conditions rechecked in fp32.

D THE INFORMATION RATCHET AND THE LEARNING TOLL

D.1 WORK EXTRACTED WITH PARTIAL KNOWLEDGE

A box in contact with a heat bath at temperature T_{B} holds one molecule in one of A equal cells. A demon holding a belief q over the cells slowly moves the walls until cell y occupies a fraction $q(y)$ of the volume. The molecule's own cell x expands isothermally from volume $1/A$ to $q(x)$, delivering $k_B T_{\mathrm{B}} \ln\big(A\, q(x)\big)$ of work drawn from the bath, and removing the walls afterwards costs nothing. This gives equation 1. Averaging over $x \sim p$ gives $\mathbb{E}[W] = k_B T_{\mathrm{B}} \big[\ln A - H(p) - D_{\mathrm{KL}}(p \,\|\, q)\big]$: the entropy of p is what no demon can collect, and the divergence is lost because the protocol was designed for q rather than p (Kelly, 1956; Vinkler et al., 2016; Kolchinsky & Wolpert, 2017).

D.2 EXACT LEARNING TOLL AND ITS ASYMPTOTIC FORM

Figure 8 compares the exact toll with the asymptotic forms of Proposition 1 on four of the eight families of §3: the three for which the constant c_π has a closed form, and the hidden-Markov family, which has none and shows what the comparison looks like without one. For the four-sided die ($\mathrm{Dir}(\mathbf{1})$ prior, $\dim = 3$), i_t is computed exactly from the Pólya-urn distribution of the counts. For linear regression ($w \sim \mathcal{N}(0, I_{\dim})$, $x \sim \mathcal{N}(0, I_{\dim})$, $\sigma^2 = 1/4$), $i_t = \mathbb{E}\,\frac{1}{2}\ln\!\big(1 + x_t^\top \Sigma_{t-1} x_t/\sigma^2\big)$ with Σ_{t-1} the posterior covariance, estimated from 50000 episodes. For the hidden-Markov family ($A = 4$, two hidden states, $\dim = 8$), i_t is the per-position loss of the Bayes mixture over a library of 32768 tasks minus that of the oracle model, averaged over 3072 episodes. The constant of the asymptotic toll is $c_\pi = \frac{3}{2}\ln\frac{1}{2\pi e} + 2\,(\psi(4) - \psi(1)) - \ln 3! = -2.38$ for the die and $c_\pi = \dim \ln 2$ for regression, both from the expression of Proposition 1 evaluated on the family—the same

Dirichlet–categorical integral that Lemma 7 does for the Rényi order two; it has no closed form for the hidden-Markov family, whose asymptote is therefore not drawn in panel b, although its exact toll is. Examples of the hidden-Markov family, here single symbols, are not i.i.d. given the task, so Proposition 1 does not cover it, and its rate $\dim/2t$ is shown for reference only. Its values are also approximate: enlarging the library from 32768 to 131072 tasks lowers the cumulative toll at $T = 256$ from 6.85 to 6.74 nats, so late hidden-Markov values are slight overestimates.

The per-example rate $\dim/2t$ overestimates the toll early in the context, by a factor of 5 at the first example of the die and 22 for the hidden-Markov family, and it meets the exact values after tens to hundreds of examples. The cumulative form $\frac{\dim}{2}\ln T + c_\pi$ is negative for the die up to $T = 4$ and overestimates the regression toll at short lengths (2.77 against 1.30 at $T = 1$, $\dim = 4$); it is within 4% of the exact toll from $T = 64$ on for the die and both regression settings.

E TRANSFORMERS TRAINED ON A FRESH TASK IN EVERY SEQUENCE

Section 3.1 draws a single conclusion across forty settings at once. A transformer trained on a fresh task in every sequence pays, after each demonstration, the learning toll that was computed from the task family before training began, and by the end of its context it has taken between 97.3 and 99.94% of the learning gain. This appendix separates those forty settings and reports each of them on its own. The first section gives the learning toll, the learning gain, the dissipation and the learning efficiency of every family at every context length. The second asks how much of the small shortfall that remains belongs to the task family and how much belongs to training that has not yet finished. The third restores the oracle model's loss, which Figure 3a subtracts away, so that the size of the toll can be judged against the loss a model actually pays. The fourth asks whether a model that predicts as the ideal in-context learner predicts also carries the belief that the ideal learner carries.

E.1 THE LEARNING EFFICIENCY OF EACH FAMILY AT EACH CONTEXT LENGTH

Figure 9 reports the four quantities of §3 for each of the forty settings separately. The columns are the eight task families and the horizontal axis of every panel is the context length, so that a single transformer stands behind each point. Every one of them was trained for 100k steps from the same seed under one protocol, and families and context lengths can therefore be compared without confounds.

Each row carries a quantity with its own unit, and the four rows are best read in order. The learning toll $\Lambda(T)$ measures what a whole context is worth, and it grows with the length of that context in

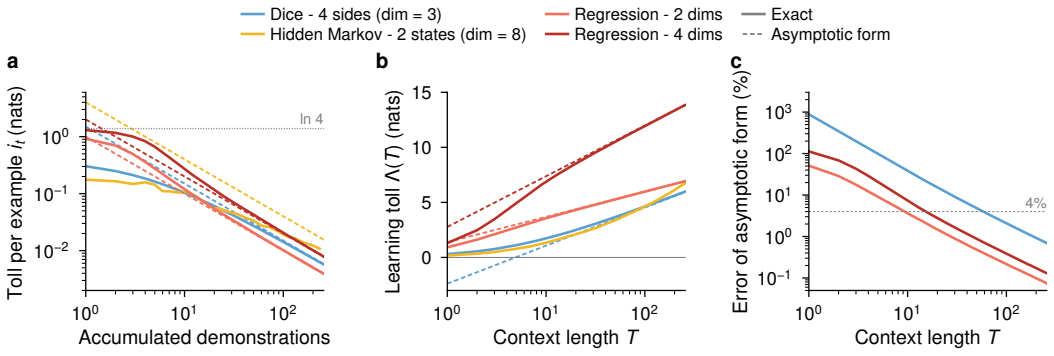

Figure 8: **Exact learning toll against its asymptotic form**, on the four-sided die, the two-state hidden-Markov family and the two regression settings. **a**, Toll paid at each example, i_t (solid), and the asymptotic rate $\dim/2t$ (dashed, same colours); dotted, the loss of a uniform guess over four symbols, which bounds i_t for the die and hidden-Markov families. Hidden-Markov values are averaged over log-spaced bins. **b**, Learning toll $\Lambda(T)$ (solid) and $\frac{\dim}{2}\ln T + c_\pi$ (dashed), drawn for the three families whose c_π has a closed form. **c**, Relative error of those three against the exact toll; the dashed line marks 4%.

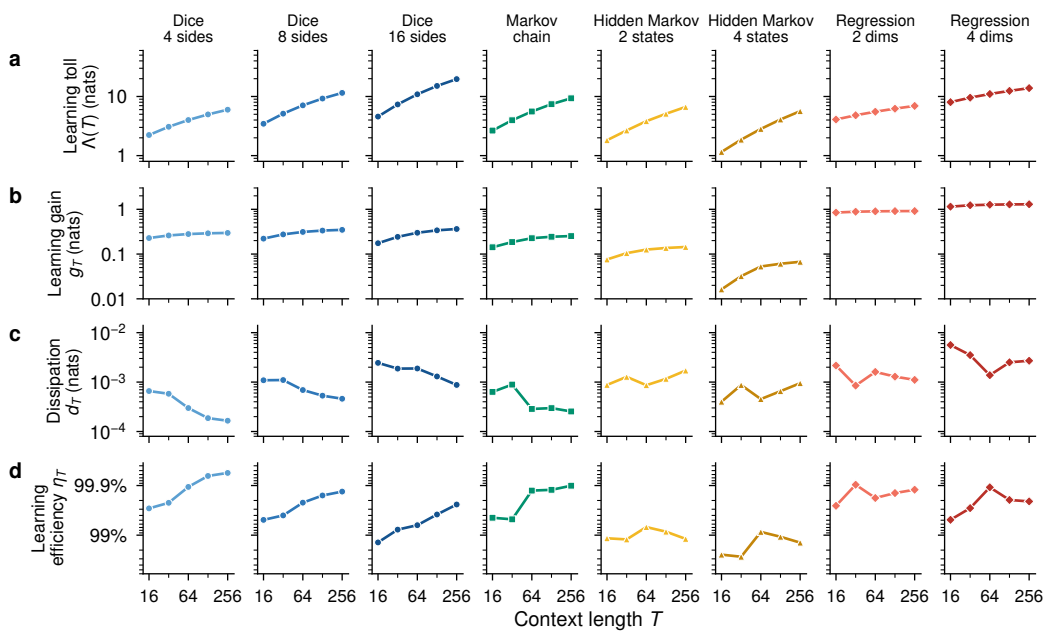

Figure 9: **The four quantities of Section 3, reported for each of the forty settings.** The columns are the eight task families and the horizontal axis of every panel is the context length. Each row has its own vertical axis. **a**, The learning toll $\Lambda(T)$, computed from the task family alone, in nats per context. **b**, The learning gain g_T, and **c**, the dissipation d_T of the trained model, both in nats per demonstration and read over the last eighth of the context. **d**, The learning efficiency that the two of them leave, $\eta_T = 1 - d_T/g_T$, drawn on the scale of Figure 3c.

every family, running from 1.15 to 19.7 nats across the sweep. The learning gain g_T measures something different, namely how much of the first demonstration's toll the context has already provided by position T, and this quantity saturates at i_1 rather than growing. On Dice - 4 sides, for instance, the gain moves only from 0.228 to 0.297 nats while the toll of the same setting grows from 2.23 to 5.98 nats. A longer context is therefore worth more in total while what each further demonstration adds falls away, and the first two rows together make that distinction concrete. The dissipation d_T that the trained model adds at the same margin remains between one and a half and three orders of magnitude below the gain, and the bottom row reports what the gain and the dissipation leave between them, the learning efficiency $\eta_T = 1 - d_T/g_T$. All forty values lie between 97.28 and 99.94%.

In six of the eight families the efficiency rises as the context grows, and the widest of these climbs is Dice - 16 sides, which moves from 98.60 to 99.76%. The rows above show why this happens, since the gain saturates while the dissipation either falls or holds steady, so that the ratio of the two improves. The two hidden-Markov families behave differently. Their efficiency peaks at $T = 64$ and falls back to 98.81 and 98.59% at $T = 256$, and the third row locates the cause, because on Hidden Markov - 2 states the dissipation rises from 0.0009 to 0.0017 nats over the same range while the gain saturates as it does everywhere else. These two are the families whose demonstrations are not independent of one another given the task, which is the condition that Proposition 1 requires, and the same distinction separates them again in §4.2, where their toll grows far more slowly than a count of the task's parameters would predict.

E.2 How far training had gone after 100k steps

The efficiencies of §3.1 are read from a single checkpoint, and a reader is entitled to ask whether the small shortfall that remains there belongs to the task family or simply to training that has not yet finished. To answer this question we evaluated the same runs every 5k steps throughout training (Figure 10).

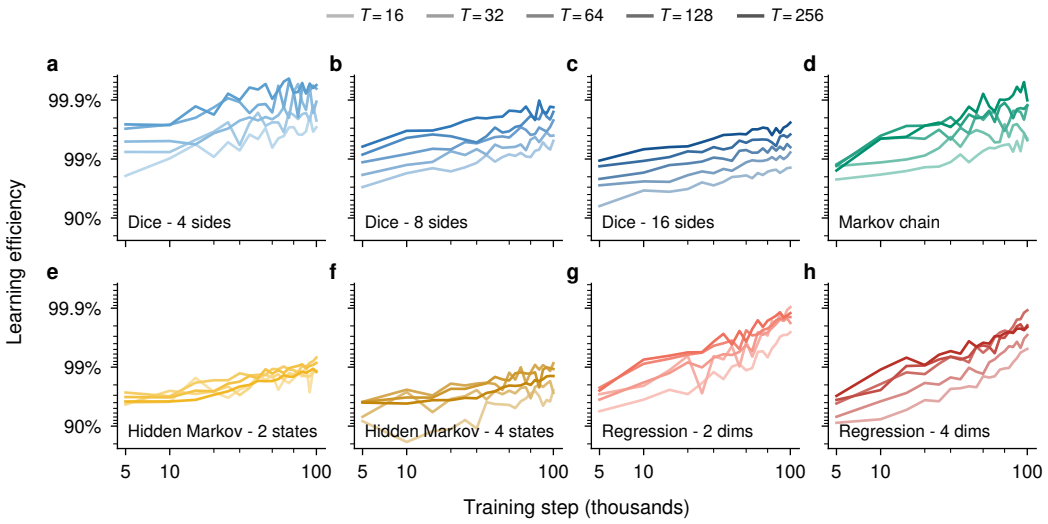

Figure 10: **Learning efficiency along training.** Each panel shows one task family and each line one context length. The efficiency is drawn on the scale of Figure 3c, and both axes are logarithmic so that the first evaluations, where most of the approach takes place, are given room. Evaluation begins at 5k steps. The untrained checkpoint, which lies between -1071 and -37%, is three orders of magnitude below this range and is omitted here; Figure 3c reports it. Thirty-two of the forty settings are still improving at 100k steps, and the regression settings are furthest from having stopped.

Before training begins, all forty settings lie between -1071 and -37%, with an average of -218%. A network of this shape therefore ends a context worse off than a predictor that never reads it at all, and the longer the context the worse it does. The first five thousand steps carry every one of the forty settings to somewhere between 91.2 and 99.7%, so that almost the whole of the distance is covered at once. The remaining 95k steps close what is left slowly, and the shortfall still present at 25k steps is a median of 2.36 times the shortfall still present at 100k.

That closing had not finished when training stopped. If a setting is counted as still improving when its remaining shortfall shrinks by more than 5% over the final 25k steps, then 32 of the 40 settings were still improving at 100k, the regression families and Dice - 8 sides fastest and the two hidden-Markov families slowest, while seven settings had a larger remaining shortfall at 100k than at 75k. Two consequences follow. First, the efficiencies reported in this paper are in most settings a lower bound on what the same architecture reaches with longer training, which makes the claim of §3.1 a conservative one. Second, the settings with long contexts are further from convergence than those with short ones, so that part of any trend along a panel of Figure 9d reflects how far training has progressed rather than a property of the family itself.

E.3 The size of the learning toll beside the loss a model pays

Figure 3a subtracts the oracle model's loss away and plots what remains, which is the comparison the toll calls for, but which leaves a reader with no sense of how large the toll is beside the loss a model actually pays. Figure 11 restores that floor for every family at a context length of 256. The toll turns out to be a small part of the total loss, and a part that shrinks as the context grows: on Dice - 4 sides it amounts to 0.29 nats at the first demonstration, against an oracle loss of 1.09 nats, and to 0.006 nats against 1.11 nats at the last. In all eight families the trained model's curve follows the ideal in-context learner's at every position, which is the same agreement that the efficiency printed in each panel expresses as a single number.

Two features of this comparison deserve comment. The first concerns the finite evaluation set. Both reference curves are computed exactly, whereas the model's curve is an average over $1{,}024$ evaluation sequences, and at individual positions the measured loss can therefore fall slightly below the ideal learner's. On Dice - 4 sides this happens at 86 of the 256 positions, by at most 1.4×10^{-3} nats. It is sampling noise rather than a violation of Theorem 1, because the dissipation is itself

a divergence between two predictive distributions evaluated on the same sequences, and such a divergence is non-negative at every sequence and every position.

The second concerns what happens when the same comparison is summed over a whole context. The ordering survives in the six families whose reference predictor is exact, by margins of 0.045 to 0.69 nats, and in Hidden Markov - 4 states by 0.022 nats. The single setting that comes out below is Hidden Markov - 2 states, at -0.027 nats against a toll of 6.66 nats, and there the reference predictor is not exact either. It is the Bayes mixture over a library of $32{,}768$ tasks, which Appendix D.2 shows to overestimate the toll at this context length. Nothing in these measurements approaches the region that Theorem 1 forbids.

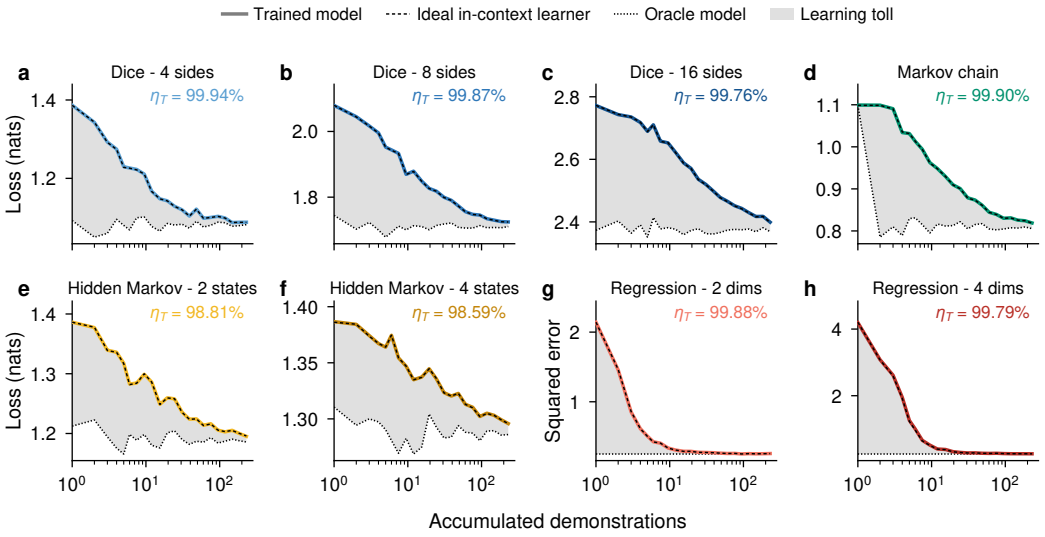

Figure 11: **The learning toll beside the loss a trained model pays**, for every family at a context length of 256. In each panel the dotted curve is the oracle model, which is told the task and still pays the noise within it; the dashed curve is the ideal in-context learner; and the shaded band between the two is the learning toll. The trained model, drawn in the family colour, coincides with the dashed curve at this scale, and the annotation gives that run's learning efficiency. **g**, **h**, The regression families, where the loss is squared error and the oracle model is the noise level σ^2. All curves are averaged over log-spaced bins of positions.

E.4 Whether the models carry the belief the ideal learner carries

Section 3.1 establishes that these models predict as the ideal in-context learner predicts. It does not establish that they hold what the ideal learner holds, which is a posterior distribution over the task, because a model could in principle arrive at the same predictions by some other route. Trained transformers are known to represent the Bayesian belief about the task linearly in their residual streams (Shai et al., 2024; Piotrowski et al., 2025), and we asked whether the models of §3.1 do the same.

The probe is a ridge regression, solved in closed form with $\lambda = 10^{-3}$, from the residual-stream activations at positions $t \in [T/4, T)$ to the Bayes posterior mean of θ given $x_{\le t}$. We average R^2 over the components of θ and report the best layer. The same probe is fitted to an untrained network of the same shape, which provides the baseline against which a trained value has to be judged.

We fitted the probe on all twenty discrete settings of the sweep (Figure 12). After training, the posterior mean of the three dice families is recovered with an R^2 between 0.90 and 0.99, which reproduces what has been reported before. What the untrained baseline adds is that a large part of this is not the work of training. Where the probe is easy the untrained network already scores 0.96 and training adds only 0.006, whereas where it is hard, on Dice - 16 sides at $T = 256$, the untrained network scores 0.29 and training adds 0.61. The gain from training therefore lands where

the baseline is weakest, and this is what the widening gap between the two curves of each panel shows.

The Markov chain is an informative exception. Its trained R^2 is the lowest of the four families, between 0.54 and 0.75, while its learning efficiency is among the highest, between 99.52 and 99.90%. Dice - 16 sides stands the other way round, with an R^2 between 0.90 and 0.95 against an efficiency as low as 98.60%. The family whose predictions come closest to the ideal is thus the family whose belief a linear probe reads least well, so that linear decodability and predictive quality do not move together. Part of this difference is the dimension of what is being decoded, since the Markov belief is a 3×3 transition matrix whereas the belief of Dice - 4 sides is a point in a three-parameter simplex, and that alone is reason enough not to read R^2 as an absolute scale for how well a belief is held. We therefore report this measurement as an exploratory one, because it says what a linear probe finds and not what the model uses.

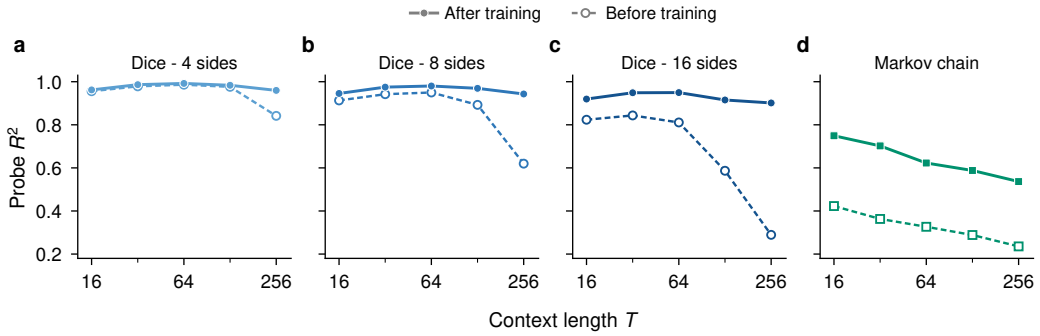

Figure 12: **What a linear probe recovers of the Bayes posterior mean.** Each panel shows one discrete task family and the horizontal axis is the context length. The vertical axis is the best-layer R^2 of a ridge probe from the residual stream to the posterior mean of the task. Filled markers joined by solid lines give the trained network, and open markers joined by dashed lines an untrained network of the same shape. The two curves separate where the probe is hard, which is to say where the alphabet is large or the context long, and they coincide where it is easy.

F Further results for Section 3.2: frozen language models against the exact floor

Section 3.2 reads thirty models on three task families. Every claim it makes is a claim about what a context is worth and where a model loses it, so every claim can be put to a wider record, and that is what this appendix does. We ask the same three questions of fifty models on six task families: do they take most of what a context offers, do they pay several times the learning toll to get there, and is what they leave learning rather than a wrong prior however the models are grouped. That frozen language models deviate from an exact Bayesian reference, and that instruction tuning can widen the deviation, is established (Falck et al., 2024; Chlon et al., 2025; Ozelim & Zenil, 2026; Sorensen et al., 2026); what an exact floor adds is that the deviation can be read in one unit and split. The models, the tapes and the coverage gate are those of Appendix C, and all values are read on the last eighth of a context of 256 demonstrations unless stated otherwise.

F.1 The measurement does not depend on the three families or the thirty models

We recorded 57 open models, from 14 million to 33 billion parameters, on the three dice, the Markov chain and the two hidden-Markov families of §3, each family written once in common letters and once, as a control for surface form, in rare ones. Fifty models clear the coverage gate on at least one family in common letters. Because the models are causal, the first T positions of each record are a context of length T, so one record gives the learning efficiency at every shorter length as well.

The picture §3.2 draws from three families holds across six (Figure 13a,b). The median model ends the context with 84.6% of the learning gain on Dice - 4 sides, 83.0% on Dice - 8 sides, 84.7% on Dice

- 16 sides, 79.6% on the Markov chain, 74.8% on Hidden Markov - 2 states and 54.3% on Hidden Markov - 4 states: most of what the context offers, never all of it, on every family. Efficiency rises with scale but never cleanly—at every size some models end below zero, worse off than a predictor that never read the context—and Llama-3.1-8B is the best model on five of the six, between 91.0 and 96.5%, with Qwen2.5-32B-Instruct the best on the Markov chain at 91.9%.

Nor does the choice of family change which models look good. Among the 36 models scored on all six, the rank correlation of the efficiencies between any two families is 0.75 to 0.99, median 0.90, and the thin lines of Figure 13b stay roughly parallel as they fall. The surface form does not change it either: on Dice - 4 sides the rare letters give the same order (rank correlation 0.92) at a median 3.4 points lower (Figure 13c), so writing the tape in unfamiliar symbols costs something but does not change what the measurement says about a model.

What does move is the model family, which is why Figure 4c is read as showing that the family matters less than scale or version rather than as a ranking of the four. Qwen2.5 gives up the least of the learning gain on all three families of §3.2—a median 7.6% on Dice - 4 sides, 9.6% on Dice - 8 sides and 13.0% on the Markov chain—but behind it the order changes: Llama is second at 10.2% on Dice - 4 sides and last at 16.7% on the Markov chain, where Mistral is second at 14.1%.

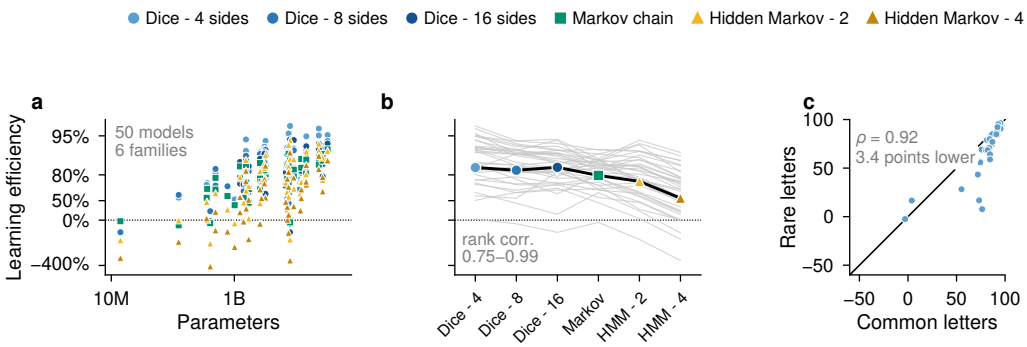

Figure 13: **Fifty frozen models on six task families.** Learning efficiency is on the inverted scale of Figure 3c, so the models that end a context worse off than one that never read it stay on the axis. **a**, Every model on every family it clears the coverage gate on, against parameter count. **b**, The same values family by family: one thin line for each of the 36 models scored on all six, the median in ink. **c**, The surface-form control on Dice - 4 sides: the same tape written in rare letters against common ones, with the identity line.

F.2 They pay several tolls, and it is the late context that is wasted

Section 3.2 reports that the average model pays 2.6 to 3.4 times the learning toll over a whole context and 4.3 to 6.4 times over its last eighth, so that most of what it still pays late is waste. Figure 14 reads the same quantity model by model on Dice - 4 sides, for the 48 that clear the gate there, and adds the transformer of §3.1 trained on this family as the one engine that does not waste.

The curves separate where §3.2 says they do (Figure 14a). The trained transformer lies on the exact toll for the whole context; the frozen models start on it and peel away one after another, and the weakest flattens out entirely. In units of the toll (Figure 14c) the median model pays 3.5 times it over the whole context, with quartiles 2.8 to 5.1, and 5.8 times it over positions 150–200, with quartiles 3.8 to 10.1: the same two numbers §3.2 gives, on a record that includes models far weaker than the thirty it uses, and the late multiple is the larger one for every model in the record.

That the excess is waste rather than the price of learning can be read directly (Figure 14b). Over positions 150–200 the exact toll is 0.0084 nats per symbol while the models' excess runs from 0.020 (Mistral-Small-24B-Base-2501) to 0.314 (Pythia-410M-deduped), so between 58 and 97% of what they pay there is dissipation, median 83%. The trained transformer pays 1.8%. The shortfall of a frozen model late in its context is almost entirely the engine's.

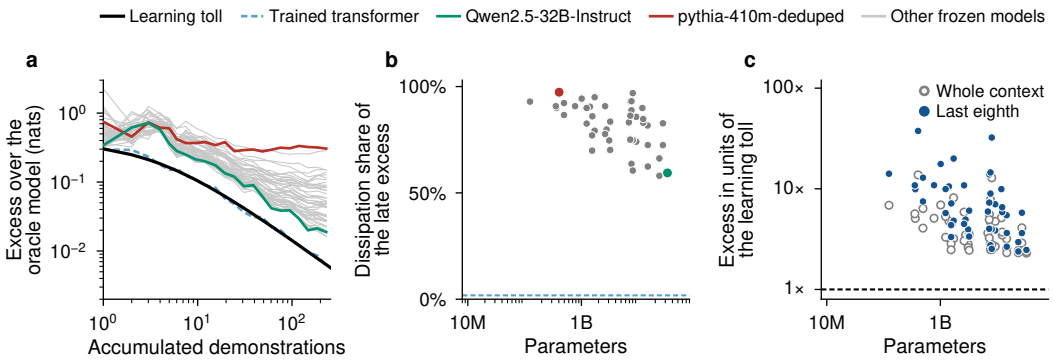

Figure 14: **What a frozen model pays against the exact floor** (Dice - 4 sides, common letters, the 48 models that clear the gate). **a**, Excess over the oracle model at each position, averaged over log-spaced bins, against the exact toll and against the transformer of §3.1 trained on this family. **b**, The share of the excess over positions 150–200 that is dissipation; the dashed line is the trained transformer. **c**, The same excess in units of the learning toll, summed over the whole context and read over positions 150–200; the dashed line is one toll.

F.3 The split survives every grouping, and a fourth one

The central claim of §3.2 is that what these models leave is learning lost rather than a prior the evidence would wash out, and that this holds however the models are grouped. Figure 15 is Figure 4b–d redrawn on the whole record: the same split, the same boxes, the same axis, over 50 models and 184 model-family pairs instead of 30 and 90, with a fourth panel that groups by task family.

Every grouping keeps the shape. The learning lost is a median 7.9 to 15.6% of the gain in every box of Figure 15a–c, and the prior lost a median 1.2% or less. It falls with scale and it is the learning part that falls: below 3B a model gives up a median 16.8% of the gain, at 3–7B 10.6%, at 7–12B 11.0% and above 12B 8.5%, while the prior lost moves only from 1.2 to 0.6%. The four model families sit close together, from 8.6% for Qwen2.5 to 14.0% for Mistral, as §3.2 reports on three of them.

Instruction tuning behaves as §3.2 describes, and the wider record makes the paired form of the claim much stronger than the grouped one. Grouped, the median rises only from 13.4 to 14.1%, because the two groups do not contain the same models. Paired against its own base, an instruction-tuned model gives up more of the gain in 52 of the 59 base-and-instruct pairs in the record, and the median pair moves the learning lost by $+3.6$ points while moving the prior lost by $+0.07$: instruction tuning takes away the updating and leaves the starting belief where it was.

The fourth panel is the one §3.2 cannot show, and it carries one caveat. On the three dice the prior lost is a median 0.1, 0.6 and 1.3% of the gain against a learning lost of 12.1, 13.6 and 13.0%, so the split reads as it does in the main text. On the Markov chain the prior lost is 6.0%—still small beside a learning lost of 12.7%, but six times what the dice give. The add-α rule fitted there is a rule over transition counts, a wider family of wrong priors than on a die, so it absorbs more of the dissipation; the prior lost is a lower bound in every family, and it is a looser one here.

G What a trained model can hold

Section 4.1 reports two limits on the switch. The first is computed from the task family before training and is the subject of Theorem 2. The second is measured after training and is the number of tasks the model itself holds, read as the pool at which φ passes $\frac{1}{2}$; it is what makes the measured switch fall short of the computed one wherever the family asks for more tasks than the model has. This appendix is about that second limit. It asks first whether the limit is really one on the weights at all, and then what moves it.

The alternative is that what binds is not the weights but the context. A network with a context memory of C nats resolves the task only as well as some $n_{\text{eff}} < T$ samples would, so what would

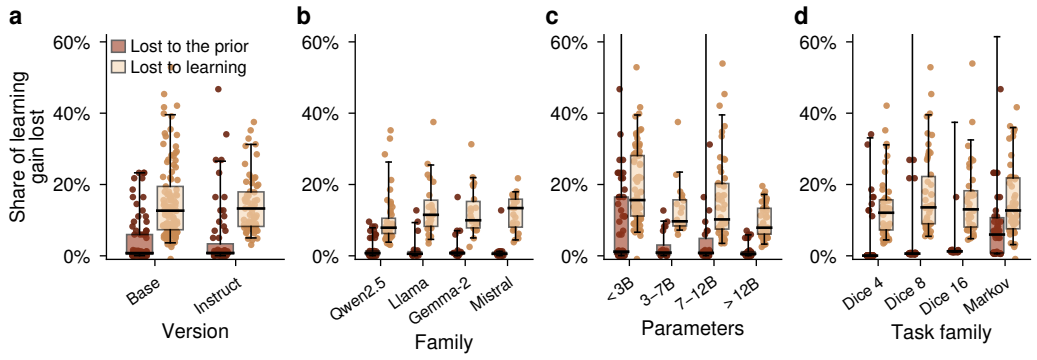

Figure 15: **Figure 4b–d on the whole record.** The share of the learning gain a model gives up, split as in equation 6, for the 50 models and four task families on which the add-α rule is defined. Boxes span the quartiles, whiskers the 5th to 95th percentiles, one point per model and family. **a**–**c**, The three groupings of Figure 4, here with every model of the record. **d**, The same split by task family.

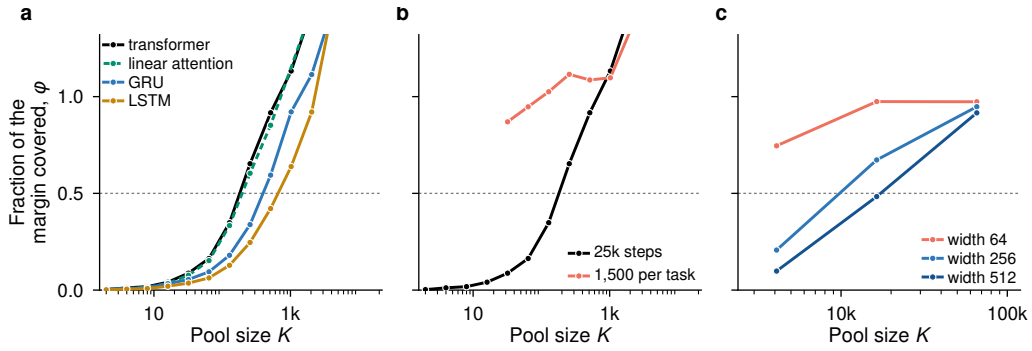

Figure 16: **What moves the number of tasks a model holds.** φ is the fraction of the retrieval margin a trained model has covered, so $\varphi = 0$ is the exact memorizer of the pool and $\varphi = 1$ is as far from it as the ideal in-context learner stands; the dashed line marks the $\varphi = \frac{1}{2}$ at which §4.1 reads off what a model holds. **a**, Four architectures on the four-sided die under one protocol (25k steps, batch 64, three seeds, $T = 128$), sharing the same pools and therefore the same denominator; linear attention is dashed because it lies on the transformer. **b**, The transformer of **a** against an arm in which every task is seen 1,500 times at every pool size, so that the step budget grows with the pool instead of the exposure falling with it. **c**, Three widths under the 200k-step protocol on the sixteen-sided die at $T = 128$.

bound the switch is the information the network can keep, $\min(\Lambda_\pi(T)$, what C nats retain), rather than what the context offers. How many samples a memory of C nats is worth is itself not settled. A representation argument, in which a belief is quantized to C nats, gives $\Lambda_\pi(n_{\text{eff}}) = C$, so that n_{eff} grows as $e^{2C/\dim}$. For a coin ($\dim$=1), time-invariant machines with $S = e^C$ states have minimax risk $\Theta(1/S)$ (Leighton & Rivest, 1986; Berg et al., 2021), so n_{eff} grows only as e^C, while machines that know the position can approach the quantization limit asymptotically. Transformers keep the whole context and know positions, so for them the context side should not bind.

We checked that directly. Recurrent networks whose state size is fixed (4 to 128 units), linear attention and softmax transformers were run at T=2048 on families with $\dim = 3$, 7 and 15, and we took a forgetting floor and a rising sensitivity to the order of the demonstrations as the two signatures of a memory limit. Neither appeared. An 8-unit GRU tracks exact Bayes to about 10^{-4} nats per token out to position 2047 at $\dim$=3, and the only floors, in the narrowest networks at $\dim$=15, are flat from the first positions and equally present in a softmax transformer of the same width, which makes them approximation floors rather than memory floors. The networks of §4

therefore keep far more of their context than their tasks require, and the limit that binds there is the other one.

What moves that limit is how the model was trained and what kind of network it is. Four architectures were trained on the four-sided die under one protocol, 25k steps at a batch of 64 with three seeds, at $T = 32$ and $T = 128$ and over the same pools (Figure 16a). Because the computed switch is read from the task family alone, it is the same number for all four, 12 tasks at $T = 32$ and 92 at $T = 128$, and so is the retrieval margin that φ divides by; the four curves differ only in how far each model stands from the memorizer. At $T = 128$ a transformer holds 181 tasks and linear attention 198, while a GRU holds 420 and an LSTM 642, a spread of a factor of three and a half. A recurrent network of the same width therefore stays a memorizer over pools at which an attention network has already become a learner.

The training budget moves the same level. Section 4.2 reports the controlled comparison, in which four times the steps at a fixed batch raises what a transformer holds from 166 to 646 tasks, a factor close to the factor of four in the budget itself. Figure 16b shows what that dependence looks like across a whole sweep. One arm spends 25k steps at every pool, so that each task is seen 50,000 times at the smallest pool and fewer as the pool grows; the other lets the step budget grow with the pool, so that every task is seen 1,500 times whatever the pool size. Under the fixed budget the transformer is an almost exact memorizer at the smallest pools and passes $\varphi = \frac{1}{2}$ at 181 tasks, where each task has been seen about 8,800 times. Under fixed exposure it never reproduces the memorizer anywhere in the range, with φ at or above 0.87 from the smallest pool on. The two arms differ in total steps as well as in exposure per task, so they do not separate those two, but together they place the level firmly on the training side: a model that receives of the order of a thousand sequences per task memorizes no pool we ran.

Width moves the level as well, and by less than a count of parameters would require (Figure 16c). Under the 200k-step protocol on the sixteen-sided die at $T = 128$, a width of 64 has already passed $\varphi = \frac{1}{2}$ below the smallest pool we ran, a width of 256 passes it at about 9,800 tasks and a width of 512 at about 17,300, a factor of 1.8 for a doubling of width. This is the conclusion §4.2 reaches from the measured switch, 0.29 nats per doubling of parameters against the 0.69 that a fixed store per parameter would need, reached here through φ instead.

Changing the architecture does not move the switch point. Figure 5b sets the two limits against each other across the settings of §4.1, and the four architectures above make the same point with the task family held fixed and only the network changed: their measured switches differ from one another by 0.03 nats at $T = 32$ and by 0.19 nats at $T = 128$, against the factor of three and a half that separates what the same models hold. The two context lengths also show when the second limit matters. At $T = 32$ the family asks for 12 tasks, far fewer than any of the four models holds, and every measured switch lies within 0.05 nats of the computed one. At $T = 128$ it asks for 92, close enough to what the models hold that the shortfall appears, and the shortfall then shrinks as the model holds more: -0.32 nats for the transformer at 181 tasks, -0.30 for linear attention at 198, -0.19 for the GRU at 420 and -0.13 for the LSTM at 642. That ordering is what a second limit predicts: the more tasks a model holds, the less its measured switch falls below the computed one.

H Further results for Section 4.3: what a pool has to be before post-training selects memorizing

Section 4.3 post-trains a pretrained model against pools of different sizes, at one context length and with every task in the pool equally likely. Theorem 2 makes both of those things variables, and it is easier to see how once the switch is written in nats rather than in tasks. The theorem fixes a pool size, $\ln K^*(T) = \Lambda_\pi(T) + O(1)$ for the categorical families of this section; a uniform pool of K tasks has entropy $H(\theta) = \ln K$, so the same statement reads $H^* = \Lambda_\pi(T) + O(1)$: a pool stops being worth memorizing once its entropy reaches the learning toll, up to that constant. Both sides are then in the same unit, and two predictions follow that §4.3 does not test. The crossing should move with the context length, because $\Lambda_\pi(T)$ does. And it should depend on the entropy of the pool rather than on the number of tasks in it, so that a pool can be made worth memorizing without a single task being removed from it.

H.1 THE CONTEXT LENGTH MOVES THE CROSSING

We post-trained Qwen2.5-0.5B at $T = 32$ and $T = 128$ over pools of 4 to 4096 tasks, and Qwen2.5-1.5B at $T = 512$ over pools of 32 to 65,536. The three learning tolls are $\Lambda_\pi = 3.07$, 4.97 and 7.00 nats, so the prediction is three crossings about two nats apart.

The first half of the test needs no model at all. Figure 17a is what a pool of a given entropy costs its own exact memorizer, computed from the task family: the three curves are one curve shifted, and they shift with the context. A longer context resolves more of the task than a small pool contains, so the same pool is a worse bargain at $T = 512$ than at $T = 32$. Reading each curve where that cost passes the one nat of c_{pen} gives $H = 3.31$, 5.56 and 7.37 nats, against learning tolls of 3.07, 4.97 and 7.00. Across pools from four tasks to sixty-five thousand, and contexts from 32 to 512 demonstrations, the crossing tracks the toll to within 0.6 nats.

The second half is measured (Figure 17b). Each sweep is flat where its pool sits above its own crossing—between 0.1 and 0.6 nats, which is what post-training on a diverse pool costs a model that was already reading its context—and climbs where the pool sits below it. The size of that climb is set by the toll and not by the pool alone: a pool of four tasks takes the model to 19.4 nats at $T = 128$ but only to 4.7 at $T = 32$, where the toll is 1.9 nats smaller and four tasks are correspondingly less of a bargain.

One caveat belongs with the figure. The three lengths are not one model swept over T: $T = 32$ and $T = 128$ are Qwen2.5-0.5B and $T = 512$ is Qwen2.5-1.5B, because that is how the runs exist. Panel a is therefore the clean test—it is computed from the task family and no model enters it—while panel b shows only that each model's measured sweep turns where its own computed curve says it should.

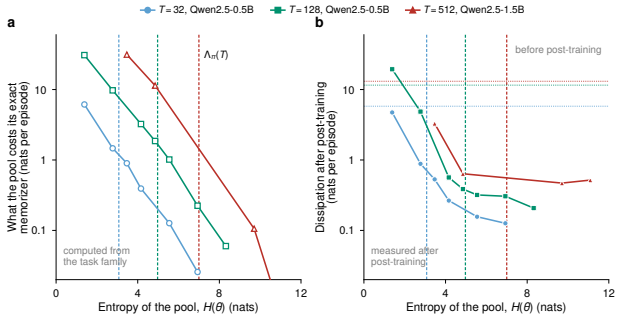

Figure 17: **A longer context asks a pool for more entropy before memorizing stops paying.** Both panels are read on the entropy of the post-training pool, $H(\theta) = \ln K$; dashed verticals are the learning toll $\Lambda_\pi(T)$ of each length. **a**, What the pool costs its own exact memorizer, computed from the task family alone. **b**, Dissipation on a new task after post-training; dotted horizontals are the same models before it.

H.2 THE DISTRIBUTION OF THE POOL, NOT THE NUMBER OF TASKS IN IT

A pool's entropy can be lowered in two ways. Tasks can be taken out of it, which is what §4.3 does. Or every task can be left in and some of them made rare, which lowers the entropy while the number of tasks never changes. Running both separates the count from the distribution, and they turn out not to be the same lever.

We post-trained Qwen2.5-0.5B at $T = 128$ on pools of exactly 4096 tasks whose sampling weights follow a Zipf law of exponent $s = 0$, 1, 1.5 and 2 (Figure 18a). That lowers $H(\theta)$ from 8.32 to 1.63 nats—below the toll of 4.97—without removing anything: all 4096 tasks are still in the pool at every setting, and the rarest are still drawn.

The count is not what decides. Skewing alone moves the post-trained model from 0.21 to 4.40 nats of dissipation on a new task, a factor of 21, with the pool held at 4096 tasks throughout (Figure 18b). A corpus can therefore be made to select against in-context learning without being made small.

The two routes are not equivalent, and the difference is in how far the model travels toward the optimum for its own pool. On the skewed sweep it arrives: its dissipation is 1.05, 1.20 and 1.22 times that of the exact memorizer of the weighted pool at $s = 1$, 1.5 and 2, so the filled and open markers of Figure 18b sit on top of one another. On the uniform sweep it stops short: 0.63, 0.50 and 0.17 times its memorizer's dissipation at $K = 4$, 16 and 64, so it stays one and a half to six times better than the predictor it is being pulled toward. Making tasks rare is the more effective way to select memorizing, not the weaker one.

We do not have a demonstration of why, and offer the reading only as a reading. A skewed pool still holds 4096 distinct tasks, so none of it can be stored outright and the pull acts through which tasks the gradient sees rather than through how many there are; a pool of four tasks, by contrast, is small enough to store outright, and there the model leaves some of the room it is given unused. What the measurement does establish is that the number of tasks in a pool does not determine what post-training selects, and that two pools of the same entropy can leave a model in different places.

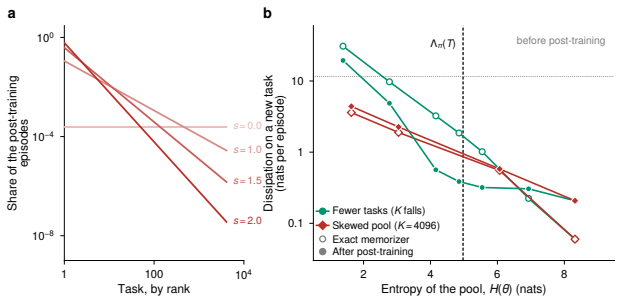

Figure 18: **Two ways to lower a pool's entropy, and only one of them takes a model all the way to its memorizer.** Qwen2.5-0.5B, $T = 128$. **a**, The sampling weights of the skewed pools: a Zipf law of exponent s over the same 4096 tasks. **b**, Dissipation on a new task against the entropy of the pool, for both sweeps; filled markers are the post-trained model, open markers the exact memorizer of that same pool, and the dashed vertical is the learning toll. The two sweeps share their right-hand endpoint, the uniform pool of 4096 tasks.

I What the measurement does not rank

Everything measured in this paper is measured on tapes whose floor is known, and every claim we make is a claim about those tapes. Whether the same number also ranks how well a model uses a context in general—the reading a user of the scale would want—is a separate question, and this appendix reports what we can say about it. The answer is negative, and we record it here rather than leave the stronger reading available.

Three measurements of context use. All three are run on frozen models already in the record of Appendix F. The first is a *bijection* task: the model is shown k demonstrations of a random assignment of four classes to four labels and must predict the label of a held-out item. The assignment is drawn afresh for every episode, so zero-shot performance is chance by construction and the entire score is the in-context gain; no baseline of general ability can contribute to it. The second is *copying*: a passage is presented twice and we measure how far the loss falls on the second copy, which is the induction-head capability in its most elementary form and again cannot be supplied by anything but the context. The third is ordinary few-shot accuracy on two real classification tasks, AG News and DBpedia, at 0, 1, 5 and 16 demonstrations. Thirty-six models were scored on the first two and twenty-eight on the third; 26 and 21 of them respectively also clear the coverage gate on Dice - 4 sides, and it is those models that can be placed on both scales.

Across models the correlation is real, and it is scale. Learning efficiency on our tape and the bijection gain correlate, $\rho = +0.57$ over the 26 models ($p = 0.002$, Figure 19a), and so do learning efficiency and 16-shot accuracy, $\rho = +0.63$ over the 21 ($p = 0.002$, Figure 19b). Neither survives the obvious control. Log parameter count on its own reaches $\rho = +0.85$ on the bijection gain and

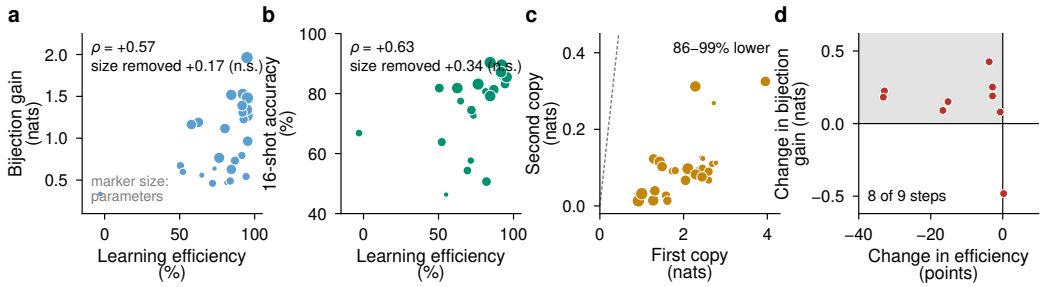

Figure 19: **Learning efficiency against three measurements of context use.** Marker size grows with parameter count. **a**, The bijection gain, whose zero-shot baseline is chance by construction, against learning efficiency on Dice - 4 sides (26 models). **b**, 16-shot accuracy on two real classification tasks (21 models). In both, the correlation and what is left of it once the trend against log parameter count is removed from both axes. **c**, Copying: the loss on the second presentation of a passage against the loss on the first, with the identity line. **d**, The within-pair control: for each of nine post-training steps, the change in learning efficiency against the change in bijection gain, with the quadrant in which the two disagree shaded.

$+0.82$ on the accuracy, so it is the better predictor of both; and once the trend against log parameter count is removed from both axes, what is left of the correlation with efficiency is $+0.17$ on the bijection task and $+0.34$ on the accuracy, neither significant at these sample sizes. A scale that agrees with parameter count, less well than parameter count does, has not been shown to measure anything parameter count does not.

The copying probe has no resolution. Every model in the record copies almost perfectly: the loss on the second copy is 86 to 99% below the loss on the first, from 0.92–3.95 nats down to 0.013–0.325 (Figure 19c). The probe therefore separates nothing, and the absolute fall in nats—the quantity one would naively read—is bounded by how badly the model did on the first copy, which itself tracks efficiency at $\rho = -0.74$. Reading the fall rather than the ratio would produce a correlation of the wrong sign for a reason that has nothing to do with context use.

Within one lineage the two orderings are opposite. The sharpest test holds size and the base model fixed and asks what a single post-training step does to both scales. Nine such steps are in the record—instruction tuning at six sizes, code specialization and reasoning distillation of Qwen2.5-7B, and instruction tuning of SmolLM2-1.7B—and eight of the nine move the two in opposite directions (Figure 19d): the learning efficiency falls, by 0.7 to 33.2 points, while the bijection gain rises, by 0.08 to 0.43 nats. Only Qwen2.5-32B-Instruct goes the other way. Whatever these steps do, they make a model worse at tracking a Bayesian posterior over a process it has been told nothing about and better at reading an arbitrary mapping out of a context, and we do not have an account of why.

What we claim instead. We therefore do not claim that dissipation on symbol tapes ranks how well models use context in general, and a reader should not take the efficiencies of Appendices E and F as a leaderboard for in-context learning. What the measurement gives is narrower and is what the rest of the paper uses it for: a training-free, absolute reading of how far a model runs from a reversible bound on a process whose floor can be computed, and the split of that distance into a prior lost and a learning lost, which no relative benchmark can provide. The reversal above is worth recording on its own account, since it separates two things the literature often treats as one.