PRETRAINING LATENT INFORMATION FEEDBACK TRANSFORMERS WITH TEACHER SUPERVISION

Anonymous authors
Paper under double-blind review

ABSTRACT

Transformer language models (LMs) are feed-forward: deep-layer representations are never fed back to shallower layers, and the only pathway for information to flow downward across generation steps is the decoded token. This narrow channel forces models to recompute intermediate results and to discard alternative continuations. In this work, we remove this bottleneck *during pretraining*, introducing the LIFT (Latent Information Feedback Transformer) architecture and training method which enable LMs to propagate state across generation. We achieve this by turning recurrent-state learning into a teacher-forced prediction problem: each input token is paired with an information-dense state, derived from the next-token distribution of an off-the-shelf pretrained LM. The model, extended with a small number of additional parameters, is then trained to predict both the next token and the next state. As the input states are precomputed, pretraining remains fully parallel across positions. At inference, the model's own predicted states are fed back, with a minor computational overhead that decreases with model size. Experiments with pretrained models ranging from 135M to 1B parameters show that LIFT consistently outperforms standard Transformers and baselines on language modeling, downstream reasoning tasks, and procedural tasks under token-matched budget, while being on par or ahead of compute-matched Transformers. Moreover, a controlled study on a state-tracking task shows that a tiny LIFT outperforms same-size Transformers trained on $8\times$ more data, even when trained with the states of a Transformer that fails the task. Overall, we show that LMs can learn to exploit deep-to-shallow feedback through an efficient and scalable pretraining procedure.

1 INTRODUCTION

Scaling language models (LMs) has been driven by major improvements to the Transformer architecture (Fedus et al., 2022; DeepSeek-AI et al., 2024; Qiu et al., 2025). Still, the design is feed-forward: deeper layer representations are never fed back to shallower ones. The only pathway through which information propagates "downward" is the decoded token. This creates a narrow communication channel that bottlenecks the computation, leading the model to recompute previous computations (Fan et al., 2021; Biran et al., 2024) or drop possible continuations (Zhu et al., 2025).

Several efforts have attempted to mitigate this bottleneck at post-training, by teaching models to generate compressed representations of their reasoning traces. During inference, these methods propagate a continuous hidden representation at each step instead of a single token (Hao et al., 2025; Shen et al., 2025; Wei et al., 2026). Although promising, such methods require expensive sequential latent rollouts during training, which may not suffice for rewiring the model to this new input distribution of richer continuous representations (Rizvi-Martel et al., 2026; Wu et al., 2026).

A natural question that arises is whether we can teach LMs to exploit such deep-to-shallow feedback *at pretraining*, while avoiding infeasible training costs resulting from sequential processing. Recent work tackles this by approximating full recurrence via Jacobi-style iterations, updating positions in parallel through multiple forward passes (Zeng et al., 2026a; Cai et al., 2026; Wang et al., 2026; Huang et al., 2026; Zhang & Ren, 2026). However, this approach has an inherent quality–efficiency tradeoff: a better approximation requires more iterations and thus more training cost.

In this work, we propose LIFT (Latent Information Feedback Transformer), an alternative approach for pretraining LMs with state propagation, which converts recurrent-state learning into a teacher-

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.

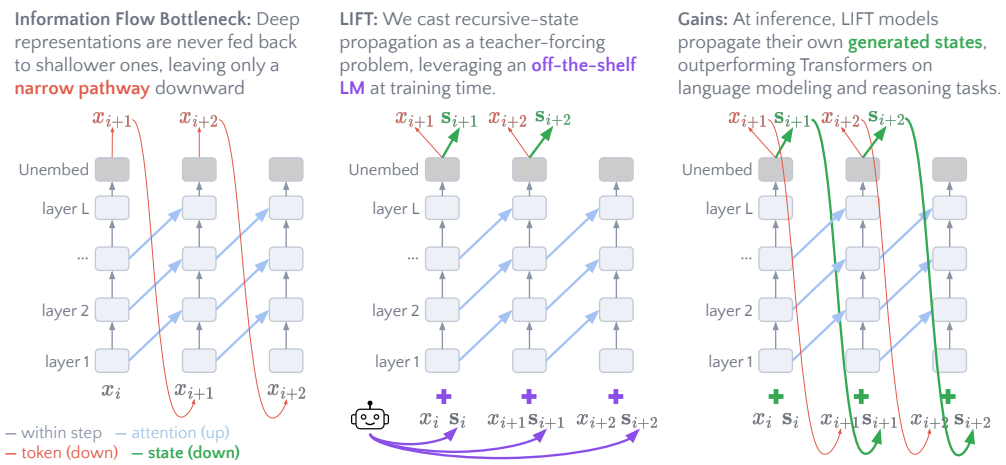


Figure 1: We remove the information flow bottleneck in Transformers (left) by introducing the LIFT architecture and pretraining approach (middle), which allows state propagation at inference (right).

forced prediction problem (Figure 1). During training, the model receives at every position a token x_i and state $\mathbf{s}_i$, and is trained to predict both the next token x_{i+1} and the next state $\mathbf{s}_{i+1}$. To keep training efficient, we capitalize on the prevalence of publicly available LMs and obtain states from *next-token distributions* of an external "teacher" model[1] which runs in parallel on the same texts. At inference, the model feeds its own predicted states back through this channel, alongside its generated tokens. Importantly, the states used in training are informative from the first training step and produced independently of the trained model, preserving parallelism across sequence positions.

We evaluate LIFT through comprehensive experiments, demonstrating its added expressivity and consistent performance gains. First, we consider S_5 permutation composition, a state-tracking task that fixed-depth Transformers cannot solve as sequence length N grows (Merrill et al., 2024). Training two-layer Transformer models on this task shows that, while a vanilla Transformer fails on sequences of $N \geq 12$ steps, LIFT models solve 12-step sequences (100% accuracy) even when the states come from a Transformer teacher that itself fails at that length (3%). Thus, the feedback channel lets a fixed-depth model solve longer sequences than a Transformer of the same depth can, and parallel training on fixed-depth Transformer states is sufficient for learning it.

Next, we pretrain LIFT LMs from 135M to 1B parameters, on up to 100B tokens at the 1B scale (about $5\times$ the Chinchilla budget; Hoffmann et al., 2022), and evaluate them on language modeling and a wide suite of downstream benchmarks. Across evaluations, LIFT outperforms token-matched Transformers at every scale, as well as distillation and Jacobi-iteration feedback baselines, and is level with or ahead of compute-matched Transformers. The largest gains are on procedural tasks, such as multi-operand arithmetic and pattern continuation, where LIFT matches or beats a Transformer trained on $2\times$ more tokens at every scale.

Overall, our work shows that Transformer LMs can learn to exploit deep-to-shallow feedback through efficient, fully parallel pretraining with teacher-forced states, improving both language modeling and reasoning. Our contributions can be summarized as follows:

- We introduce, to our knowledge, the first method for pretraining a Transformer LM with a teacher-forced latent feedback channel: a pretrained LM supplies the states during training, keeping training fully parallel, and the model generates its own states at inference.
- We develop LIFT, which uses next-token distributions as states, aligning state supervision with the model's native prediction task.
- On a synthetic state-tracking task, we show that the feedback channel lets a fixed-depth LIFT solve the task at lengths where a Transformer of the same depth fails.

[1]Unlike in knowledge distillation (Hinton et al., 2015), the teacher's role here is to supply the student's *inputs*, not only its targets; at inference the teacher is removed and the student consumes its own states.

- We demonstrate the advantages of LIFT LMs of 135M to 1B parameters, which achieve better language modeling capabilities and downstream performance in token-matched and compute-matched settings, compared to a standard Transformer and to feedback variants.

2 Information Flow Bottleneck in Transformer LMs

Consider an L-layer autoregressive Transformer LM, p_θ, with hidden dimension d and vocabulary $\mathcal{V}$ (Vaswani et al., 2017). For an input sequence of tokens $x_1, \ldots, x_n$, let $\mathbf{h}_i^\ell \in \mathbb{R}^d$ denote the representation at position i after layer ℓ, with $\mathbf{h}_i^0$ being the input token embedding. At each position, the output projection $U \in \mathbb{R}^{|\mathcal{V}| \times d}$ produces the next-token distribution:

$$\mathbf{p}_i = \mathrm{softmax}(\mathbf{o}_i) \;\; ; \;\; \mathbf{o}_i = U\mathbf{h}_i^L, \tag{1}$$

where $\mathbf{o}_i$ are the logits. At inference, information moves across positions via two channels:

$$\begin{aligned} \text{Attention channel:} \quad & \mathbf{h}_{i \le j}^\ell \rightarrow \mathbf{h}_j^{\ell+1} \\ \text{Token feedback channel:} \quad & \mathbf{h}_i^L \rightarrow \mathbf{p}_i \rightarrow x_{i+1} \rightarrow \mathbf{h}_{i+1}^0. \end{aligned} \tag{2}$$

Communication *upward* is mediated by attention, which lets later positions $j > i$ read the representation at a previous position i. Token feedback is the only way to move information *downward*; a representation can influence a shallower hidden representation in subsequent steps *only* at decoding time, and only by encoding information in the current position's output, which re-enters as input in the next step. Crucially, this channel is extremely narrow, limited to a single decoded token x_{i+1}, which can carry at most $\log_2 |\mathcal{V}|$ bits of information (≈ 17 bits for $|\mathcal{V}| = 100{,}000$).

Allowing lower layers to access previously computed higher-layer representations, i.e., $\mathbf{h}_i^L \rightarrow \mathbf{h}_{i+1}^0$, could potentially enhance the model's computation. First, later positions could reuse results computed in deep layers. Without such access, an intermediate result that emerges deep in the model is out of reach of lower layers that need it, so the model must recompute it earlier or fail (Fan et al., 2021). For instance, when a model resolves the first hop of a multi-hop query "too late" for its remaining layers to complete the second, routing that late representation back into an earlier layer corrects up to 66% of the failures (Biran et al., 2024). Second, the model could carry several continuations forward, rather than committing to the sampled token. This matters when the model must explore alternatives: in graph reachability, continuous thoughts that hold many search paths at once let a two-layer Transformer find the answer in as many steps as the graph's diameter, instead of following a single path, which requires quadratically more steps (Zhu et al., 2025). Third, the computation's depth could grow with the sequence, as in a recurrent network. This matters for tasks whose number of sequential steps grows with the input, such as state tracking over a long sequence, which fixed-depth Transformers cannot solve (Merrill et al., 2024; Merrill & Sabharwal, 2025).

3 Latent Information Feedback Transformers (LIFT)

Motivated by the problem above, we propose LIFT, a scalable architecture and training method for deep-to-shallow state propagation across positions, effectively removing the information flow bottleneck (Figure 1). We describe the modified architecture that fuses states with input tokens (§3.1), and explain our approach for parallel training with teacher-forced states (§3.2).

3.1 LIFT Architecture for Next Token-State Prediction

Our goal is to let LMs propagate an additional state alongside the predicted token, carrying more information from deep layers. LIFT implements this by predicting a state along with every token:

$$x_{i+1}, \mathbf{s}_{i+1} = p_\theta(\, x_{\le i}, \mathbf{s}_{\le i}) \tag{3}$$

Token and state prediction In LIFT models, both x_{i+1} and $\mathbf{s}_{i+1}$ are derived from the model's output logits $\mathbf{o}_i$. The next token x_{i+1} is sampled as usual (Eq. 1), while $\mathbf{s}_{i+1}$ is obtained by truncating the output distribution to its top-k tokens: the k largest logits are kept, the rest are set to $-\infty$, and a softmax with temperature τ renormalizes over the kept tokens:

$$\begin{aligned} \mathbf{o}_i' &= \mathrm{top}_k(\mathbf{o}_i) \\ \mathbf{s}_{i+1} &= \mathrm{softmax}(\mathbf{o}_i'/\tau) \in \mathbb{R}^{|\mathcal{V}|}, \end{aligned} \tag{4}$$

so that $\mathbf{s}_{i+1}$ has at most k non-zero entries. Building the state from the model's output distribution lets the model carry substantially more information than the sampled token: it ranks the alternative continuations and retains much of the context that produced it (Morris et al., 2024).

Notably, a possible alternative for states could be the model's hidden representations before the output projection (Hao et al., 2025; Shen et al., 2025; Cai et al., 2026). We use distributions as they are the more natural choice in our case, where the model learns to predict the teacher's states (§3.2). For any two models that share a tokenizer, these distributions live in the same space, where coordinates correspond to token probabilities, and predicting them is a well-established objective (Hinton et al., 2015). The student's own predictions, which replace the teacher's states at inference, then live in that same space too. Hidden states have no shared coordinates across models (Sriram et al., 2018), so the student would have to regress onto another model's internal representation, which need not align with its own and which, in our early exploration, we found harder to predict.

Token and state fusion To process the joint input of tokens and states, we design a dedicated *fusion* layer that transforms them into a single representation, which feeds the standard Transformer blocks. Specifically, $\mathbf{s}_i$ is first projected through the token embedding matrix $E \in \mathbb{R}^{|\mathcal{V}|\times d}$, yielding the empirical mean induced by $\mathbf{s}_i$, and then concatenated with $\mathbf{h}_i^0$. The concatenation is fused through a SwiGLU block (Shazeer, 2020) with a residual connection to the token embedding:

$$\begin{aligned} \mathbf{u}_i &= \left[\mathrm{RMSNorm}(\mathbf{h}_i^0)\ ;\ \mathrm{RMSNorm}(W_s\, E^\top \mathbf{s}_i)\right] \\ \hat{\mathbf{h}}_i^0 &= \mathbf{h}_i^0 + W_d\big(\mathrm{SiLU}(W_g \mathbf{u}_i) \odot W_u \mathbf{u}_i\big) \end{aligned} \tag{5}$$

where $W_s \in \mathbb{R}^{d\times d}$, $W_g, W_u \in \mathbb{R}^{2d\times 2d}$ and $W_d \in \mathbb{R}^{d\times 2d}$ are the fusion layer's parameters. At the first position, where no previous output exists, the projected state $E^\top \mathbf{s}_1$ is replaced by a learned vector. The fusion layer holds all the parameters LIFT adds: $11d^2$ weights and the initial-state vector, 3% of our 1B model. Since the Transformer stack grows with Ld^2, this share shrinks with depth. The fused representation $\hat{\mathbf{h}}_i^0$ is then fed to the standard Transformer stack, interacting with the context, which in turn contains information about previous tokens and states by the same mechanism.

3.2 LIFT Training with Teacher-Forced States

Decoding in autoregressive Transformers is already sequential, so feeding back the state (Eq. 3) adds only a minor cost. However, at training time, such recurrent computations forfeit the parallelism that enables scalable training of Transformer LMs. To tackle this, we observe that this dependency disappears if the input states are computed *in advance*: instead of obtaining them via autoregressive generation, the states can be supplied by an external teacher, through *teacher forcing*. This reduces the problem to finding a teacher capable of producing informative states. Here, we capitalize on the growing availability of open pretrained LMs (Wolf et al., 2020), which offers many capable teachers, recycling their pre-invested compute into a higher-quality training signal.

Training with teacher states Given a training sequence $\mathbf{x} = \langle x_1, \ldots, x_n\rangle$, the teacher first processes $\mathbf{x}$ in parallel, emitting for every position i its own output logits $\mathbf{o}_i^\star$. These logits are then transformed according to Eq. 4 to yield teacher states $\mathbf{s}_{i+1}^\star$. Notably, as the teacher is frozen, it admits common training optimizations that require fixed weights (e.g., Xiao et al., 2023; Aminabadi et al., 2022). Once the states are all available, they are paired with their matching tokens, i.e., $(x_{i+1}, \mathbf{s}_{i+1}^\star)$. The pairs are then passed to the model's fusion layer *in parallel* (Eq. 5), and forwarded through the full backbone as in standard causal-LM training, *without* propagating gradients to the teacher. The state channel is thus teacher-forced, same as the token channel, and full parallelism is maintained.

At inference, states are produced by the model autoregressively, making input processing sequential. To allow prompt prefilling in one parallel pass without states, we employ *prefix state dropout* during training: the states of each training sequence are replaced with probability p by a learned bias $\mathbf{b} \in \mathbb{R}^d$ (§A.1). The model thus learns to operate both with and without states, allowing prefilling.

Mitigating train–inference state distribution shifts Because the model is trained on teacher states but consumes its own states at inference, a mismatch between the two distributions can accumulate over decoding steps (Ranzato et al., 2016). We mitigate this in two ways. First, we supervise the model's state predictions with the teacher's, so that the states it generates at inference remain

close to those it was trained on. To this end, we add a state-alignment term to the training objective:

$$\mathcal{L} = \sum_i \Big(\underbrace{-\log \mathbf{p}_i(x_{i+1})}_{\text{language modeling}} + \lambda \underbrace{\mathcal{D}_k\big(\mathbf{o}_i^\star \,\|\, \mathbf{o}_i\big)}_{\text{state alignment}} \Big), \tag{6}$$

where $\lambda \geq 0$ weights the two terms and $\mathcal{D}_k$ is a forward-KL divergence (Hinton et al., 2015) adapted to top-k distributions (§A.2).

Second, in the last 10% of training, we adapt the model to its own states, dropping the teacher and the alignment term. Each training step executes one or two initial forward passes (with probability 50%), to produce the model's own states: the first pass reads the input tokens with the bias $\mathbf{b}$ at every position. The second (optional) pass is fed the states predicted by the first. To train the model, we pass the states produced at the end of the process above instead of those typically supplied by the teacher, without propagating gradients back to the initial iterations (further details are in §A.3).

4 Evaluation of State Tracking Capabilities

We start by evaluating LIFT on state tracking, showing the expressive advantage of its architecture over a standard Transformer, and how it can learn from both strong and weak teachers.

The S_5 task In this task, a model must predict the composition of a sequence of permutations, which requires state tracking along the sequence (Liu et al., 2023). Let S_5 be the group of permutations of five elements (120 in total). Given an initial state $s_0 \in S_5$ and permutations $a_1, \ldots, a_N \in S_5$, the model must predict the final state s_N obtained by applying them to s_0:

$$s_N = a_N \circ \cdots \circ a_1 \circ s_0, \qquad s_i = a_i \circ s_{i-1}. \tag{7}$$

We encode all the elements in S_5 as single tokens; thus, the model is given the sequence of tokens $\langle s_0, a_1, \ldots, a_N \rangle$ and should predict a single token. Importantly, this task is NC^1-complete: unless $\mathsf{TC}^0 = \mathsf{NC}^1$, a Transformer of fixed depth cannot solve it for growing N, and the number of layers must grow as $\Omega(\log N)$ (Merrill & Sabharwal, 2025). Conversely, a recurrent model needs only a constant number of layers, as its computation depth grows with the sequence (Merrill et al., 2024).

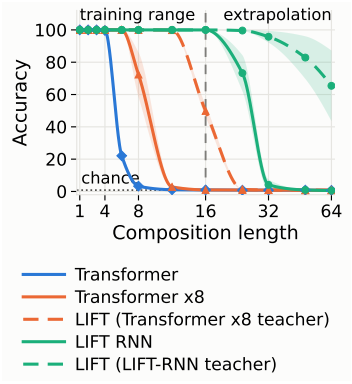


Figure 2: S_5 accuracy at increasing task lengths N (linear for $N \leq 16$, logarithmic for $N > 16$), of Transformers (solid lines) versus LIFT models (dashed lines). Standard error is over seeds.

Experimental setup We train two-layer Transformers with 2M parameters from scratch; LIFT adds its fusion layer (7% of the parameters). Input sequences of lengths $N \in [1, 16]$ are drawn from 12 fixed generators of S_5 at every training step, with a random initial state. We report accuracy on 2,000 held-out sequences per length, up to $N{=}64$, averaged accuracy over 3 seeds (see §B.1 for more details). We compare the following models:

1. **Transformer**: A token-matched vanilla Transformer, trained for $5{,}000$ steps. Following Liu et al. (2023), the loss is computed on the correct state s_i at every position i, not only on the final s_N. At test time, it answers in one parallel pass.
2. **Transformer$\times$8**: Same as the above baseline, except that the model is trained $8\times$ longer ($40{,}000$ steps). Notably, larger training budgets do not yield substantial improvements (see §C.1).
3. **LIFT-RNN**: The LIFT architecture trained sequentially for $40{,}000$ steps: at every position the output distribution is embedded with the token and passed to the next position, and gradients flow across steps using only the answer loss.
4. **LIFT (Transformer$\times$8 teacher)**: Trained in parallel for $5{,}000$ steps with this teacher.
5. **LIFT (RNN teacher)**: Trained in parallel for $5{,}000$ steps with LIFT-RNN as a teacher.

Results Figure 2 shows that the fixed-depth Transformer struggles with this task, as theory predicts: even with $8\times$ the training budget, it solves only lengths up to $N{=}6$ and is at chance from

N=12 on. Training sequentially with latent feedback (LIFT-RNN) removes this bottleneck, allowing a model of the same depth to solve every length in the training range. When trained fully in parallel on the states of this recurrent model, LIFT inherits its unbounded computation depth: it matches the teacher in the training range and even extrapolates beyond it, achieving 96% at N=32 and 65% at N=64, although trained only on lengths up to N=16. Interestingly, LIFT shows such extrapolation even when trained with a depth-bounded non-recurrent teacher. LIFT solves every length up to N=12 and half of the sequences at N=16, whereas its teacher, a same-size Transformer trained with 8× the budget, is at chance beyond N=8. We stress that, trained this way, LIFT does not acquire unbounded depth and fails beyond the training range. Still, learning from the teacher's states yields a large gain over a standard Transformer, even one trained much longer.

5 Experiments with Pretrained LMs

We conduct comprehensive evaluations of LIFT LMs in two setups, benchmarking against vanilla Transformer and distillation baselines on a wide set of benchmarks (§5.2), and comparing with feedback Transformers that approximate recurrent states through repeated parallel passes (§5.3).

5.1 Experimental Setting

Pretrained LMs We pretrain LIFT LMs with 135M, 350M and 1B parameters, using the OLMo 2 architecture, codebase and pretraining data (Walsh et al., 2025), each for 5× its Chinchilla token budget (Hoffmann et al., 2022). Models at 135M scale are trained with three seeds, and larger models with one seed (due to compute constraints). The teacher at each scale is a same-size model pretrained for 10× its Chinchilla budget. LIFT uses k=1024, τ=1.5 and λ=1.5 at every scale; see §D for ablations and §B.2 for additional implementation details. We compare against the following baselines, all evaluated after an identical learning-rate annealing phase (Wen et al., 2025):

1. **Vanilla Transformer**: A standard OLMo 2 Transformer, identical to LIFT except for the fusion layer. We report results for a model trained on the same token budget and on the same compute budget as ours, accounting for the teacher's forward pass (§F.1).
2. **Distillation**: The vanilla Transformer trained with the forward-KL distillation term (Hinton et al., 2015) in Eq. 6, isolating the effect of the distillation objective from that of the latent feedback.
3. **LIFT w/o states**: Our trained LIFT LM with the channel "switched off": in place of a state, the fusion layer receives the learned bias $\mathbf{b}$ of prefix state dropout (§3.2). No state is fed back, so the model runs as a standard Transformer in one parallel pass.

Comparison with feedback Transformers We adapt the setup of T^{2}MLR (Cai et al., 2026): a Transformer whose layers read a state computed by deeper layers at earlier positions, pretrained by refining all positions in parallel over repeated passes (Jacobi iterations) instead of sequentially. All models share the SmolLM2-135M backbone (Allal et al., 2025) and are trained on a 10B-token FineWeb-Edu sample (Penedo et al., 2024). We compare LIFT with **T^{2}MLR**, both our own training run and the published checkpoint. In addition, we compare with the concurrent models of Wang et al. (2026), which feed the hidden state at the previous position back to the input and train with a few passes per token. At the time of writing, the official code and models are not publicly available, so we implemented the method ourselves, using the schedule of the authors' larger runs; we call this adaptation **Multi-pass Transformer**. We also compare against a vanilla Transformer. Models are compared under (a) matched tokens, each trains on about 10B tokens, and (b) matched compute, each receives the training FLOPs of T^{2}MLR's 10B-token run, so cheaper methods train on more tokens, revisiting the data once exhausted. See §B.3 for implementation details.

Evaluation benchmarks We use the following evaluations for both sets of experiments:

1. **Language modeling**: Perplexity on held-out text of the training corpus: for the three-scale experiments, the 1M-token C4 (Raffel et al., 2020) validation subset of the OLMo 2 evaluation set (Walsh et al., 2025); for the feedback comparison, a held-out FineWeb-Edu sample of 127K tokens. Perplexity on the other sources of the OLMo 2 evaluation set is reported in §C.
2. **Downstream tasks**: Twenty benchmark suite run through OLMES (Gu et al., 2025), covering the base-model evaluation families of OLMo 2 and OLMo 3 (Team Olmo et al., 2025). Eight of

these are at chance or floor (under 3%) for all models at these scales; the other twelve form three groups, each reported as the mean over its benchmarks (the full list is in §B.4):

- *MC*: Multiple-choice questions on science, commonsense and world knowledge. Each answer option is scored by its likelihood given the question and the highest wins.
- *Gen.*: Generative tasks on reading comprehension, knowledge recall and multi-step reasoning. The model generates the answer, scored by F1 or exact match against the reference.
- *RC*: Reference-completion tasks in science, knowledge, math and code. The reference answer is scored by its likelihood under the model, reported in bits-per-byte (BPB), lower is better.

3. **Arithmetic**: Our own synthetic task, testing multi-step computation carried in the model's latent state (no chain of thought). Each sample adds and subtracts 2–10 single-digit numbers, e.g. `2 + 0 - 6 + 7 + 5 =` (answer `8`); we report the correct answer's BPB (more metrics in §B.4).

5.2 Evaluating Language Modeling, Knowledge, and Reasoning Capabilities

LIFT outperforms Transformer baselines at matched tokens and at matched compute Table 1 presents the main results, with a per-benchmark breakdown in §C. At matched tokens, LIFT is ahead on every metric at every scale: perplexity drops by 5–5.5%, and it wins 33 of the 36 downstream benchmark comparisons. At matched compute, where the Transformer trains on over 40% more data, LIFT keeps the lead on every metric at 350M and 1B and is level with or ahead of it at 135M. These gains are consistent over the three 135M-scale training runs (see §C.4). Importantly, the gains come from the added channel rather than from the training signal alone: LIFT w/o states, the same weights run without the channel, and the distillation baseline, trained with the same state-alignment objective but without the channel, trail LIFT on perplexity and on most of the benchmarks at every scale. Further ablations reinforce this (§D): training LIFT with a narrower channel, without the state-alignment objective, or without the teacher at all significantly erodes the gains.

LIFT's token advantage grows with training Figure 3b compares the cross-entropy on the evaluation set at matched checkpoints along training, showing how many more training tokens the Transformer baseline needs to reach the performance of LIFT. At every scale, this multiplier grows along training: the longer both models train, the more tokens the Transformer needs to catch up, suggesting that the token efficiency of LIFT increases with training.

The largest gains are on procedural tasks Across all evaluations, LIFT achieves the largest gains on procedural tasks—our arithmetic task and the pattern continuation task from the OLMES basic-skills suite—which require a step-by-step answer derivation. On arithmetic, LIFT is ahead of the Transformer baseline at every scale, outperforming its own teacher, which was trained on 2× more tokens (1–5% fewer bits on average; §C.5). These gains persist across increasing numbers of operands (Figure 3a). Pattern continuation asks for the next element of a short sequence that follows a rule, over numbers, letters, symbols or dates (e.g., `2 4 6 8` → `10`, `E G I K` → `M`). LIFT beats the token-matched Transformer by 6–10% in bits per byte at every scale, and the 2×-token Transformer by 4–5% at 350M and 1B (a tie at 135M; §C.6).

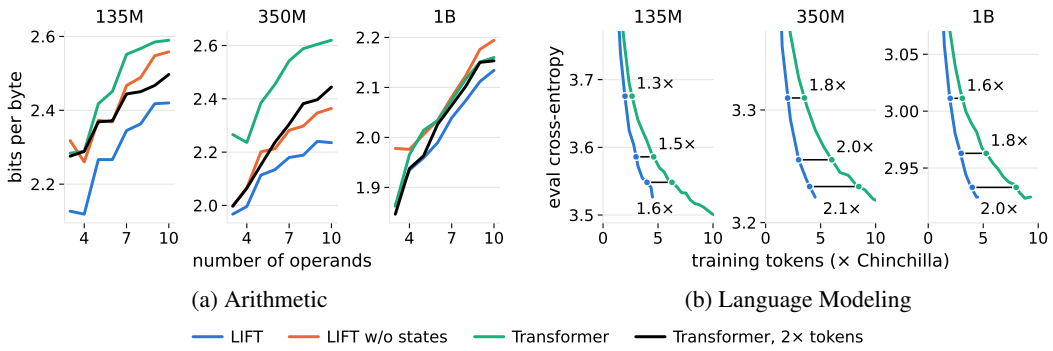
Performance with parallel prefill LIFT models can process the input prompt sequentially (with state predictions) or in parallel (with the bias state; LIFT w/o states in Table 1). The results show that LIFT w/o states is comparable to the token-matched Transformer baseline across all scales, indicating LIFT maintains strong capabilities without sequential processing of the input. In §E, we compare additional methods for performing parallel prefill with LIFT models, which vary in efficiency and quality. We find that iteratively refining the prompt, as in the adaptation phase (§3.2), recovers much of the gain of sequential processing within a few iterations.

5.3 Comparison with Jacobi-Iteration Feedback Transformers

Table 2 shows the average results over three training seeds, with seed noise and additional results reported in §C.7. At matched tokens, LIFT shows the best average performance on every metric, tied with T^{2}MLR on generation tasks, though we note that in this setting all methods are within noise. At matched compute, LIFT is on par with or better than the vanilla Transformer on all tasks, outperforming the latent feedback baselines. On the arithmetic task, performance is within the noise estimate compared to our reproduction of T^{2}MLR, though this is mainly due to the relatively high

Table 1: Performance of LIFT LMs against non-feedback Transformer baselines at three scales.

	Model	Tokens	PPL ↓	Arith. BPB ↓	MC acc. %	Gen. F1/EM %	RC BPB ↓
135M	LIFT	13.4B	**30.65**	**2.29**	39.4	**15.5**	**1.078**
	LIFT w/o states	13.4B	32.44	2.42	38.7	12.0	1.105
	Transformer, token-match	13.4B	32.43	2.46	38.1	13.6	1.110
	Transformer, compute-match	19.3B	31.04	2.50	**39.5**	14.1	1.080
	Distillation	13.4B	31.77	2.36	39.1	12.6	1.089
350M	LIFT	34.9B	**23.82**	**2.12**	**45.0**	**23.2**	**0.921**
	LIFT w/o states	34.9B	24.96	2.21	44.2	21.0	0.945
	Transformer, token-match	34.9B	25.15	2.46	43.2	21.0	0.956
	Transformer, compute-match	49.9B	24.32	2.33	44.1	22.0	0.942
	Distillation	34.9B	24.52	2.23	44.5	22.1	0.932
1B	LIFT	107.4B	**18.81**	1.99	**52.5**	**35.0**	**0.772**
	LIFT w/o states	107.4B	19.62	2.05	51.2	32.9	0.791
	Transformer, token-match	107.4B	19.79	2.03	50.5	32.9	0.808
	Transformer, compute-match	152.4B	19.17	2.07	51.6	34.8	0.792
	Distillation	107.4B	19.40	**1.95**	52.0	34.8	0.781



(a) Arithmetic (b) Language Modeling

Figure 3: LIFT performance on arithmetic and language modeling. **(a)** LIFT shows a consistent advantage on arithmetic that persists as task complexity increases, outperforming a baseline trained on 2× more tokens. **(b)** Throughout training, LIFT is more token-efficient than a vanilla Transformer, and the advantage *grows* with training: each horizontal segment marks the additional tokens the Transformer needs to reach the same loss.

variance of the latter. Notably, the feedback baselines fall behind the vanilla Transformer on every metric (except arithmetic) once compute is matched.

Crucially, pretraining with multiple iterations introduces a tradeoff between approximation quality and training efficiency. More iterations improve state approximation, providing a better training signal per token, but they make each token more expensive to process (Cai et al., 2026). Our results reflect this tradeoff: with 16 refinement passes, T^{2}MLR achieves lower perplexity than the Multi-pass Transformer at matched tokens, but at matched compute, their order reverses as the Multi-pass Transformer's fewer passes let it train on more tokens. LIFT obtains its training states from a teacher, avoiding this iterative approximation, and is equal to or better than both methods on every metric under either budget. Multi-pass training also increases memory use because activations from the passes used for backpropagation must be stored. LIFT uses less peak memory than both feedback baselines at the measured batch sizes (see §F.2 for memory usage comparison).

6 Related Work

Latent feedback in post-training A growing body of work attempts to post-train LMs to use forms of latent feedback, typically focusing on reasoning tasks. Hao et al. (2025) train LMs to reason by treating the last hidden state as a latent token, processed for a fixed number of steps. This approach, and several follow-up efforts that adapt the learning signal (Shen et al., 2025; Wei et al.,

Table 2: Comparison with Jacobi-iteration feedback Transformers at matched tokens and matched compute. We report results for T^{2}MLR using the official checkpoint (*"HF"*) and our reproduction.

	Token-matched 10B tokens						**Compute-matched** 2.3×10^{19} FLOPs					
Model	FLOPs	PPL ↓	Arith. ↓	MC %	Gen. %	RC ↓	Tokens	PPL ↓	Arith. ↓	MC %	Gen. %	RC ↓
LIFT	1.44e19	**19.42**	**2.18**	**36.7**	**7.8**	**1.705**	16.2B	18.04	**2.18**	38.0	**9.5**	1.629
Transformer	1.02e19	20.30	2.35	36.2	6.7	1.761	23.0B	**18.02**	2.30	**38.1**	8.5	**1.622**
T^{2}MLR (our run)	2.32e19	19.92	2.27	36.5	**7.8**	1.737	10.0B	19.92	2.27	36.5	7.8	1.737
T^{2}MLR (HF)	2.35e19	21.55	2.38	35.3	6.5	1.816	10.1B	21.55	2.38	35.3	6.5	1.816
Multi-pass Transformer	1.31e19	20.40	2.25	36.3	6.5	1.773	17.8B	18.58	2.25	37.6	7.7	1.675

2026), rely on sequential rollouts that are difficult to scale. To reduce this cost, Wu et al. (2025) replace the sequential rollouts with Jacobi iterations. Rather than replacing tokens with latents, Tang & Lu (2026) keep the decoded token and route the previous token's higher-layer states to lower layers, fine-tuning the model with sequential processing over groups of tokens. Alternative approaches feed back soft tokens, the probability-weighted average of token embeddings, similar to LIFT. These soft tokens are used either without any training (Zhang et al., 2025; Zhuang et al., 2025) or with reinforcement learning (Butt et al., 2026). Finally, some works teacher-force the latent inputs using supervision from gold reasoning traces (Tan et al., 2025; Amos et al., 2026) or from an oracle (Gozeten et al., 2026); this enables parallel training but requires annotated reasoning.

Latent feedback in pretraining More related to our work are methods focusing on latent feedback during pretraining. Early approaches compute the states with exact recurrence, which makes training sequential over positions (Fan et al., 2021) or over segments (Bulatov et al., 2022). More recent feedback LMs instead approximate Jacobi iterations, following fixed-point iteration methods developed to parallelize RNN training (Lim et al., 2024; Danieli et al., 2026). These LMs feed back either keys and values (Zhang & Ren, 2026) or hidden states (Zeng et al., 2026a; Huang et al., 2026; Cai et al., 2026; Wang et al., 2026), and trade training cost for approximation quality, a tradeoff we examine in §5.3. In all of these methods, states are self-generated at training time, requiring sequential rollouts or an approximation, unlike LIFT, which is the only method that teacher-forces the states with a trained LM. In Kumar & Isola (2026), an RNN is similarly pretrained with teacher-forced memory, but using an encoder trained in tandem. Tack et al. (2026) use a pretrained teacher, but only as a source of targets: the model predicts concepts extracted from it and mixes them into its hidden states. We provide an extended discussion and detailed comparison in §G.

7 Conclusion

We introduce LIFT, a Transformer LM enhanced with deep-to-shallow feedback during pretraining, realized by a state constructed from the next-token output distribution. Unlike prior methods that rely on expensive sequential rollouts or state approximations, LIFT leverages an external teacher model to provide informative states, supporting parallel training. During pretraining, LIFT is aligned to predict its own states, removing the reliance on the teacher at inference time. In a controlled setting on an S_5 state-tracking task, a two-layer LIFT successfully solves the task at lengths where its own teacher, trained for $8\times$ longer, fails to exceed chance performance. Through extensive LM pretraining experiments at the 135M–1B parameter scale, we show LIFT outperforms a standard Transformer on a variety of tasks, when matched in tokens and compute, and displays training token efficiency that grows as training progresses. Compared against alternative feedback methods, LIFT is comparable to or better than all baselines considered in both token and compute matched setting.

Notably, our work focuses on a certain choice for the teacher that shares the architecture of the student. Studying the effects of teachers with other sizes and possibly different vocabularies is an attractive avenue for future work. Another is training LIFT far beyond its teacher's training budget, a regime in which its performance and training dynamics remain open questions.

AI use statement

The authors of this work used AI tools to assist with the code design and implementation, to help with debugging, to help with writing, to improve the clarity and readability of the manuscript and

identify language errors, to create graphs and figures, and to search for relevant literature. The core of this research, including questions, hypotheses, and methodological ideas, were all developed by the authors. The AI-assisted code was reviewed and tested by the authors, and the resulting experiments and outputs were also verified. The authors take full responsibility for the integrity and accuracy of the work presented in this paper.

Ethics statement

Our work studies an architecture that provides language models with an additional non-textual channel for propagating intermediate computation across decoding steps. While such mechanisms can improve the computational capabilities and efficiency of language models, they may also reduce the extent to which intermediate computation is directly observable through generated text. This consideration relates to broader concerns regarding the potential loss of oversight as AI systems increasingly rely on internal or non-textual computation (Taylor et al., 2026). We therefore believe that the development of such architectures should be accompanied by careful evaluation of their transparency and oversight properties, and strongly support responsible and transparent research into methods for monitoring and interpreting latent computation.

Reproducibility statement

We describe our method LIFT in detail in §3 and §A. For the pretraining experiments, §B.2 specifies the exact model configurations (Table 3), optimizer, learning-rate schedule, LIFT hyperparameters and the construction of the pretraining corpus from the public OLMo 2 mix. Details specific to each experiment are given in the appendix sections referenced in its description. Upon publication, we will publicly release our code for training and evaluating all models and for producing every result in the paper, including the list of corpus files each run trains on and instructions for downloading all external resources. We will also release all the models we trained on the Hugging Face Hub.

References

Loubna Ben Allal, Anton Lozhkov, Elie Bakouch, Gabriel Martín Blázquez, Guilherme Penedo, Lewis Tunstall, Andrés Marafioti, Agustín Piqueres Lajarín, Hynek Kydlíček, Vaibhav Srivastav, Joshua Lochner, Caleb Fahlgren, Xuan-Son Nguyen, Ben Burtenshaw, Clémentine Fourrier, Haojun Zhao, Hugo Larcher, Mathieu Morlon, Cyril Zakka, Colin Raffel, Leandro von Werra, and Thomas Wolf. SmolLM2: When smol goes big — data-centric training of a fully open small language model. In *Second Conference on Language Modeling*, 2025. URL https://openreview.net/forum?id=3JiCl2A14H.

Reza Yazdani Aminabadi, Samyam Rajbhandari, Ammar Ahmad Awan, Cheng Li, Du Li, Elton Zheng, Olatunji Ruwase, Shaden Smith, Minjia Zhang, Jeff Rasley, and Yuxiong He. DeepSpeed-Inference: Enabling efficient inference of Transformer models at unprecedented scale. In *SC22: International Conference for High Performance Computing, Networking, Storage and Analysis*, pp. 1–15. IEEE, November 2022. doi: 10.1109/SC41404.2022.00051.

Ido Amos, Avi Caciularu, Mor Geva, Amir Globerson, Jonathan Herzig, Lior Shani, and Idan Szpektor. Latent reasoning with supervised thinking states, 2026. URL https://arxiv.org/abs/2602.08332.

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and Charles Sutton. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*, 2021.

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. Scheduled sampling for sequence prediction with recurrent neural networks. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015. URL https://proceedings.neurips.cc/paper_files/paper/2015/file/e995f98d56967d946471af29d7bf99f1-Paper.pdf.

Eden Biran, Daniela Gottesman, Sohee Yang, Mor Geva, and Amir Globerson. Hopping too late: Exploring the limitations of large language models on multi-hop queries. In Yaser Al-Onaizan,

Mohit Bansal, and Yun-Nung Chen (eds.), *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 14113–14130, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.781. URL https://aclanthology.org/2024.emnlp-main.781/.

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. PIQA: Reasoning about physical commonsense in natural language. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05):7432–7439, April 2020. ISSN 2159-5399. doi: 10.1609/aaai.v34i05.6239. URL https://ojs.aaai.org/index.php/AAAI/article/view/6239.

Aydar Bulatov, Yury Kuratov, and Mikhail Burtsev. Recurrent memory Transformer. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 11079–11091. Curran Associates, Inc., 2022. doi: 10.52202/068431-0805. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/47e288629a6996a17ce50b90a056a0e1-Paper-Conference.pdf.

Collin Burns, Pavel Izmailov, Jan Hendrik Kirchner, Bowen Baker, Leo Gao, Leopold Aschenbrenner, Yining Chen, Adrien Ecoffet, Manas Joglekar, Jan Leike, Ilya Sutskever, and Jeffrey Wu. Weak-to-strong generalization: Eliciting strong capabilities with weak supervision. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 4971–5012. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/burns24b.html.

Dan Busbridge, Amitis Shidani, Floris Weers, Jason Ramapuram, Etai Littwin, and Russell Webb. Distillation scaling laws. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 5977–6045. PMLR, 13–19 Jul 2025. URL https://proceedings.mlr.press/v267/busbridge25a.html.

Natasha Butt, Ariel Kwiatkowski, Ismail Labiad, Julia Kempe, and Yann Ollivier. Soft tokens, hard truths. In C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust (eds.), *International Conference on Learning Representations*, volume 2026, pp. 114650–114675, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/ba9f3bcc11c65c9591a563254e863c10-Paper-Conference.pdf.

Ziyang Cai, Xingyu Zhu, Yihe Dong, Yinghui He, and Sanjeev Arora. T^2MLR: Transformer with temporal middle-layer recurrence, 2026. URL https://arxiv.org/abs/2607.15178.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. BoolQ: Exploring the surprising difficulty of natural yes/no questions. In Jill Burstein, Christy Doran, and Thamar Solorio (eds.), *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 2924–2936, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1300. URL https://aclanthology.org/N19-1300/.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? Try ARC, the AI2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Federico Danieli, Pau Rodriguez, Miguel Sarabia, Xavier Suau, and Luca Zappella. ParaRNN: Unlocking parallel training of nonlinear RNNs for large language models. In C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust (eds.), *International Conference on Learning Representations*, volume 2026, pp. 53548–53579, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/57d8ebf4c2f050a6485f370d47656a9e-Paper-Conference.pdf.

DeepSeek-AI et al. DeepSeek-V3 technical report, 2024. URL https://arxiv.org/abs/2412.19437.

Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Łukasz Kaiser. Universal Transformers. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=HyzdRiR9Y7.

Dheeru Dua, Yizhong Wang, Pradeep Dasigi, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. DROP: A reading comprehension benchmark requiring discrete reasoning over paragraphs. In Jill Burstein, Christy Doran, and Thamar Solorio (eds.), *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 2368–2378, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1246. URL https://aclanthology.org/N19-1246/.

Angela Fan, Thibaut Lavril, Edouard Grave, Armand Joulin, and Sainbayar Sukhbaatar. Addressing some limitations of transformers with feedback memory, 2021. URL https://arxiv.org/abs/2002.09402.

William Fedus, Barret Zoph, and Noam Shazeer. Switch Transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 23(120):1–39, 2022. URL http://jmlr.org/papers/v23/21-0998.html.

Guhao Feng, Bohang Zhang, Yuntian Gu, Haotian Ye, Di He, and Liwei Wang. Towards revealing the mystery behind chain of thought: A theoretical perspective. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 70757–70798. Curran Associates, Inc., 2023. doi: 10.52202/075280-3100. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/dfc310e81992d2e4cedc09ac47eff13e-Paper-Conference.pdf.

Tommaso Furlanello, Zachary Lipton, Michael Tschannen, Laurent Itti, and Anima Anandkumar. Born again neural networks. In Jennifer Dy and Andreas Krause (eds.), *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 1607–1616. PMLR, 10–15 Jul 2018. URL https://proceedings.mlr.press/v80/furlanello18a.html.

Jonas Geiping, Sean McLeish, Neel Jain, John Kirchenbauer, Siddharth Singh, Brian Bartoldson, Bhavya Kailkhura, Abhinav Bhatele, and Tom Goldstein. Scaling up test-time compute with latent reasoning: A recurrent depth approach. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pp. 41340–41391. Curran Associates, Inc., 2025. doi: 10.52202/085713-1380. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/3b01972cf31e6fa0fe29e4b8b5c2a0a1-Paper-Conference.pdf.

Gemma Team. Gemma 2: Improving open language models at a practical size, 2024. URL https://arxiv.org/abs/2408.00118.

Fabian Gloeckle, Badr Youbi Idrissi, Baptiste Rozière, David Lopez-Paz, and Gabriel Synnaeve. Better & faster large language models via multi-token prediction. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 15706–15734. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/gloeckle24a.html.

Alperen Gozeten, Muhammed Ildiz, Xuechen Zhang, Hrayr Harutyunyan, Ankit Singh Rawat, and Samet Oymak. Continuous chain of thought enables parallel exploration and reasoning. In C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust (eds.), *International Conference on Learning Representations*, volume 2026, pp. 82881–82914, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/85ec26ef94c3acb4c195e905df1ff4f7-Paper-Conference.pdf.

Yuling Gu, Oyvind Tafjord, Bailey Kuehl, Dany Haddad, Jesse Dodge, and Hannaneh Hajishirzi. OLMES: A standard for language model evaluations. In Luis Chiruzzo, Alan Ritter, and Lu Wang (eds.), *Findings of the Association for Computational Linguistics: NAACL 2025*, pp. 5020–5048, Albuquerque, New Mexico, April 2025. Association for Computational Linguistics. ISBN 979-8-89176-195-7. doi: 10.18653/v1/2025.findings-naacl.282. URL https://aclanthology.org/2025.findings-naacl.282/.

Shibo Hao, Sainbayar Sukhbaatar, DiJia Su, Xian Li, Zhiting Hu, Jason E Weston, and Yuandong Tian. Training large language models to reason in a continuous latent space. In *Second Conference on Language Modeling*, 2025. URL https://openreview.net/forum?id=Itxz7S4Ip3.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. In *International Conference on Learning Representations*, 2021a. URL https://openreview.net/forum?id=d7KBjmI3GmQ.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset. In J. Vanschoren and S. Yeung (eds.), *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks*, volume 1, 2021b. URL https://datasets-benchmarks-proceedings.neurips.cc/paper_files/paper/2021/file/be83ab3ecd0db773eb2dc1b0a17836a1-Paper-round2.pdf.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network, 2015. URL https://arxiv.org/abs/1503.02531.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Thomas Hennigan, Eric Noland, Katherine Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karén Simonyan, Erich Elsen, Oriol Vinyals, Jack Rae, and Laurent Sifre. An empirical analysis of compute-optimal large language model training. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 30016–30030. Curran Associates, Inc., 2022. doi: 10.52202/068431-2176. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/c1e2faff6f588870935f114ebe04a3e5-Paper-Conference.pdf.

Zeyi Huang, Xuehai He, LiLiang Ren, Yiping Wang, Baolin Peng, Hao Cheng, Shuohang Wang, Pengcheng He, Jianfeng Gao, Yong Jae Lee, and Yelong Shen. Latent recurrent Transformer: Architecture exploration, training strategies, and scaling behavior, 2026. URL https://arxiv.org/abs/2605.26797.

Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In Regina Barzilay and Min-Yen Kan (eds.), *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1601–1611, Vancouver, Canada, July 2017. Association for Computational Linguistics. doi: 10.18653/v1/P17-1147. URL https://aclanthology.org/P17-1147/.

Yoon Kim and Alexander M. Rush. Sequence-level knowledge distillation. In Jian Su, Kevin Duh, and Xavier Carreras (eds.), *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pp. 1317–1327, Austin, Texas, November 2016. Association for Computational Linguistics. doi: 10.18653/v1/D16-1139. URL https://aclanthology.org/D16-1139/.

Akarsh Kumar and Phillip Isola. Pretraining recurrent networks without recurrence, 2026. URL https://arxiv.org/abs/2606.06479.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. Natural Questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:452–466, 2019. doi: 10.1162/tacl_a_00276. URL https://aclanthology.org/Q19-1026/.

Zhiyuan Li, Hong Liu, Denny Zhou, and Tengyu Ma. Chain of thought empowers Transformers to solve inherently serial problems. In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun (eds.), *International Conference on Learning Representations*, volume 2024, pp. 11911–11943, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/3309b4112c9f04a993f2bbdd0274bba1-Paper-Conference.pdf.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun (eds.), *International Conference on Learning Representations*, volume 2024, pp. 39578–39601, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/aca97732e30bcf1303bc22ac3924fd16-Paper-Conference.pdf.

Yi Heng Lim, Qi Zhu, Joshua Selfridge, and Muhammad Firmansyah. Parallelizing non-linear sequential models over the sequence length. In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun (eds.), *International Conference on Learning Representations*, volume 2024, pp. 55334–55360, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/f3bfbd65743e60c685a3845bd61ce15f-Paper-Conference.pdf.

Bingbin Liu, Jordan T. Ash, Surbhi Goel, Akshay Krishnamurthy, and Cyril Zhang. Transformers learn shortcuts to automata. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=De4FYqjFueZ.

Will Merrill and Ashish Sabharwal. A little depth goes a long way: The expressive power of log-depth Transformers. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pp. 95315–95339. Curran Associates, Inc., 2025. doi: 10.52202/085713-3186. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/88dd7aa6979e352fda7c4952ca8eac59-Paper-Conference.pdf.

William Merrill and Ashish Sabharwal. The expressive power of Transformers with chain of thought. In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun (eds.), *International Conference on Learning Representations*, volume 2024, pp. 7690–7706, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/1f59721c106ea80f613299039112f651-Paper-Conference.pdf.

William Merrill, Jackson Petty, and Ashish Sabharwal. The illusion of state in state-space models. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 35492–35506. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/merrill24a.html.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. Can a suit of armor conduct electricity? A new dataset for open book question answering. In Ellen Riloff, David Chiang, Julia Hockenmaier, and Jun'ichi Tsujii (eds.), *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 2381–2391, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1260. URL https://aclanthology.org/D18-1260/.

Tsvetomila Mihaylova and André F. T. Martins. Scheduled sampling for Transformers. In Fernando Alva-Manchego, Eunsol Choi, and Daniel Khashabi (eds.), *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop*, pp. 351–356, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-2049. URL https://aclanthology.org/P19-2049/.

John X. Morris, Wenting Zhao, Justin Chiu, Vitaly Shmatikov, and Alexander Rush. Language model inversion. In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun (eds.), *International Conference on Learning Representations*, volume 2024, pp. 35863–35883, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/999fcab97007ebef0cda9949550b4a9e-Paper-Conference.pdf.

Guilherme Penedo, Hynek Kydlíček, Loubna Ben Allal, Anton Lozhkov, Margaret Mitchell, Colin Raffel, Leandro Von Werra, and Thomas Wolf. The FineWeb datasets: Decanting the web for the finest text data at scale. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 30811–30849. Curran Associates, Inc., 2024. doi: 10.52202/079017-0970. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/370df50ccfdf8bde18f8f9c2d9151bda-Paper-Datasets_and_Benchmarks_Track.pdf.

Zihan Qiu, Zekun Wang, Bo Zheng, Zeyu Huang, Kaiyue Wen, Songlin Yang, Rui Men, Le Yu, Fei Huang, Suozhi Huang, Dayiheng Liu, Jingren Zhou, and Junyang Lin. Gated attention for large language models: Non-linearity, sparsity, and attention-sink-free. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pp. 100092–100118. Curran Associates, Inc., 2025. doi: 10.52202/085713-3345. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/904e89bb4e632e75fb47f093b620b257-Paper-Conference.pdf.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67, 2020. URL http://jmlr.org/papers/v21/20-074.html.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. SQuAD: 100,000+ questions for machine comprehension of text. In Jian Su, Kevin Duh, and Xavier Carreras (eds.), *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pp. 2383–2392, Austin, Texas, November 2016. Association for Computational Linguistics. doi: 10.18653/v1/D16-1264. URL https://aclanthology.org/D16-1264/.

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. Sequence level training with recurrent neural networks. In *International Conference on Learning Representations*, 2016. URL https://arxiv.org/abs/1511.06732.

Siva Reddy, Danqi Chen, and Christopher D. Manning. CoQA: A conversational question answering challenge. *Transactions of the Association for Computational Linguistics*, 7:249–266, 2019. doi: 10.1162/tacl_a_00266. URL https://aclanthology.org/Q19-1016/.

Michael Rizvi-Martel, Guillaume Rabusseau, and Marius Mosbach. The illusion of superposition? A principled analysis of latent thinking in language models. In *Third Conference on Language Modeling*, 2026. URL https://arxiv.org/abs/2604.06374.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. WinoGrande: An adversarial Winograd schema challenge at scale. *Communications of the ACM*, 64(9):99–106, August 2021. ISSN 1557-7317. doi: 10.1145/3474381.

Maarten Sap, Hannah Rashkin, Derek Chen, Ronan Le Bras, and Yejin Choi. Social IQa: Commonsense reasoning about social interactions. In Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan (eds.), *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*

(EMNLP-IJCNLP), pp. 4463–4473, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1454. URL https://aclanthology.org/D19-1454/.

Nikunj Saunshi, Nishanth Dikkala, Zhiyuan Li, Sanjiv Kumar, and Sashank J. Reddi. Reasoning with latent thoughts: On the power of looped Transformers. In Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu (eds.), *International Conference on Learning Representations*, volume 2025, pp. 14855–14881, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/2676109d49d1eb26d6bc584a8f556305-Paper-Conference.pdf.

Noam Shazeer. GLU variants improve transformer, 2020. URL https://arxiv.org/abs/2002.05202.

Zhenyi Shen, Hanqi Yan, Linhai Zhang, Zhanghao Hu, Yali Du, and Yulan He. CODI: Compressing chain-of-thought into continuous space via self-distillation. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 677–693, Suzhou, China, November 2025. Association for Computational Linguistics. ISBN 979-8-89176-332-6. doi: 10.18653/v1/2025.emnlp-main.36. URL https://aclanthology.org/2025.emnlp-main.36/.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling LLM test-time compute optimally can be more effective than scaling parameters for reasoning. In Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu (eds.), *International Conference on Learning Representations*, volume 2025, pp. 10131–10165, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/1b623663fd9b874366f3ce019fdfdd44-Paper-Conference.pdf.

Anuroop Sriram, Heewoo Jun, Sanjeev Satheesh, and Adam Coates. Cold fusion: Training Seq2Seq models together with language models. In *Interspeech 2018*, pp. 387–391, 2018. doi: 10.21437/Interspeech.2018-1392.

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc Le, Ed Chi, Denny Zhou, and Jason Wei. Challenging BIG-bench tasks and whether chain-of-thought can solve them. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki (eds.), *Findings of the Association for Computational Linguistics: ACL 2023*, pp. 13003–13051, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-acl.824. URL https://aclanthology.org/2023.findings-acl.824/.

Jihoon Tack, Jack Lanchantin, Jane Dwivedi-Yu, Andrew Cohen, Ilia Kulikov, Janice Lan, Shibo Hao, Yuandong Tian, Jason E Weston, and Xian Li. LLM pretraining with continuous concepts. In C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust (eds.), *International Conference on Learning Representations*, volume 2026, pp. 54214–54231, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/59056767478c7df64e6250eadfeb0a04-Paper-Conference.pdf.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. CommonsenseQA: A question answering challenge targeting commonsense knowledge. In Jill Burstein, Christy Doran, and Thamar Solorio (eds.), *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 4149–4158, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1421. URL https://aclanthology.org/N19-1421/.

Wenhui Tan, Jiaze Li, Jianzhong Ju, Zhenbo Luo, Ruihua Song, and Jian Luan. Think silently, think fast: Dynamic latent compression of LLM reasoning chains. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pp. 4646–4668. Curran Associates, Inc., 2025. doi: 10.52202/085713-0164. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/0706261aedab63814a2b73c32564b4c4-Paper-Conference.pdf.

Mohan Tang and Sidi Lu. Turbo Connection: Reasoning as information flow from higher to lower layers. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=pLIJx2cFib.

Jordan Taylor, Max Heitmann, Ed Fage, Thomas Read, and Joseph Bloom. Loss of oversight: How AI systems may become harder to audit, monitor, and investigate. Technical report, UK AI Security Institute, May 2026.

Team Olmo, Allyson Ettinger, Amanda Bertsch, Bailey Kuehl, David Graham, David Heineman, Dirk Groeneveld, Faeze Brahman, Finbarr Timbers, Hamish Ivison, Jacob Morrison, Jake Poznanski, Kyle Lo, Luca Soldaini, Matt Jordan, Mayee Chen, Michael Noukhovitch, Nathan Lambert, Pete Walsh, Pradeep Dasigi, Robert Berry, Saumya Malik, Saurabh Shah, Scott Geng, Shane Arora, Shashank Gupta, Taira Anderson, Teng Xiao, Tyler Murray, Tyler Romero, Victoria Graf, Akari Asai, Akshita Bhagia, Alexander Wettig, Alisa Liu, Aman Rangapur, Chloe Anastasiades, Costa Huang, Dustin Schwenk, Harsh Trivedi, Ian Magnusson, Jaron Lochner, Jiacheng Liu, Lester James V. Miranda, Maarten Sap, Malia Morgan, Michael Schmitz, Michal Guerquin, Michael Wilson, Regan Huff, Ronan Le Bras, Rui Xin, Rulin Shao, Sam Skjonsberg, Shannon Zejiang Shen, Shuyue Stella Li, Tucker Wilde, Valentina Pyatkin, Will Merrill, Yapei Chang, Yuling Gu, Zhiyuan Zeng, Ashish Sabharwal, Luke Zettlemoyer, Pang Wei Koh, Ali Farhadi, Noah A. Smith, and Hannaneh Hajishirzi. Olmo 3, 2025. URL https://arxiv.org/abs/2512.13961.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

Evan Pete Walsh, Luca Soldaini, Dirk Groeneveld, Kyle Lo, Shane Arora, Akshita Bhagia, Yuling Gu, Shengyi Huang, Matt Jordan, Nathan Lambert, Dustin Schwenk, Oyvind Tafjord, Taira Anderson, David Atkinson, Faeze Brahman, Christopher Clark, Pradeep Dasigi, Nouha Dziri, Allyson Ettinger, Michal Guerquin, David Heineman, Hamish Ivison, Pang Wei Koh, Jiacheng Liu, Saumya Malik, William Merrill, Lester James Validad Miranda, Jacob Morrison, Tyler Murray, Crystal Nam, Jake Poznanski, Valentina Pyatkin, Aman Rangapur, Michael Schmitz, Sam Skjonsberg, David Wadden, Christopher Wilhelm, Michael Wilson, Luke Zettlemoyer, Ali Farhadi, Noah A. Smith, and Hannaneh Hajishirzi. 2 OLMo 2 furious (COLM's version). In *Second Conference on Language Modeling*, 2025. URL https://openreview.net/forum?id=2ezugTT9kU.

Xi Wang, Ziyang Cai, Zheng Zhan, Harry Dong, Ying Fan, Gustavo de Rosa, Tim Pearce, and John Langford. Full-bandwidth transformer, 2026. URL https://arxiv.org/abs/2608.08888.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=1PL1NIMMrw.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc V Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 24824–24837. Curran Associates, Inc., 2022. doi: 10.52202/068431-1800. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/9d5609613524ecf4f15af0f7b31abca4-Paper-Conference.pdf.

Xilin Wei, Xiaoran Liu, Yuhang Zang, Xiaoyi Dong, Yuhang Cao, Jiaqi Wang, Xipeng Qiu, and Dahua Lin. SIM-CoT: Supervised implicit chain-of-thought. In C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust (eds.), *International Conference on Learning Representations*, volume 2026, pp. 56721–56742,

2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/5d087955ee13fe9a7402eedec879b9c3-Paper-Conference.pdf.

Kaiyue Wen, Zhiyuan Li, Jason Wang, David Hall, Percy Liang, and Tengyu Ma. Understanding warmup-stable-decay learning rates: A river valley loss landscape view. In Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu (eds.), *International Conference on Learning Representations*, volume 2025, pp. 42840–42885, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/6a1fe80a9e2dcda0b3e5fd0fd87eb097-Paper-Conference.pdf.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. Transformers: State-of-the-art natural language processing. In Qun Liu and David Schlangen (eds.), *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pp. 38–45, Online, October 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-demos.6. URL https://aclanthology.org/2020.emnlp-demos.6/.

Haoyi Wu and Kewei Tu. Layer-condensed KV cache for efficient inference of large language models. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 11175–11188, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.602. URL https://aclanthology.org/2024.acl-long.602/.

Haoyi Wu, Zhihao Teng, and Kewei Tu. Parallel continuous chain-of-thought with Jacobi iteration. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 914–926, Suzhou, China, November 2025. Association for Computational Linguistics. ISBN 979-8-89176-332-6. doi: 10.18653/v1/2025.emnlp-main.47. URL https://aclanthology.org/2025.emnlp-main.47/.

Junhong Wu, Jinliang Lu, Zixuan Ren, Gangqiang Hu, Zhi Wu, Dai Dai, and Hua Wu. LLMs are single-threaded reasoners: Demystifying the working mechanism of Soft Thinking. In C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust (eds.), *International Conference on Learning Representations*, volume 2026, pp. 4533–4548, 2026. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/0895de150660b6c2aef9b0f098cb1dcd-Paper-Conference.pdf.

Guangxuan Xiao, Ji Lin, Mickael Seznec, Hao Wu, Julien Demouth, and Song Han. SmoothQuant: Accurate and efficient post-training quantization for large language models. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 38087–38099. PMLR, 23–29 Jul 2023. URL https://proceedings.mlr.press/v202/xiao23c.html.

Zhenrui Yue, Bowen Jin, Huimin Zeng, Honglei Zhuang, Zhen Qin, Jinsung Yoon, Lanyu Shang, Jiawei Han, and Dong Wang. Hybrid latent reasoning via reinforcement learning. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pp. 5501–5530. Curran Associates, Inc., 2025. doi: 10.52202/085713-0195. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/087f7678af2abbe0c37ecc30bd7d1adf-Paper-Conference.pdf.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. HellaSwag: Can a machine really finish your sentence? In Anna Korhonen, David Traum, and Lluís Màrquez (eds.), *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 4791–4800, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1472. URL https://aclanthology.org/P19-1472/.

Boyi Zeng, He Li, Shixiang Song, Yixuan Wang, Zitong Wang, Ziwei He, Xinbing Wang, and Zhouhan Lin. PonderLM-2: Pretraining LLM with latent thoughts in continuous space. In *Forty-third International Conference on Machine Learning*, 2026a. URL https://openreview.net/forum?id=yVFxjNzCQm.

Boyi Zeng, Shixiang Song, Siyuan Huang, Yixuan Wang, He Li, Ziwei He, Xinbing Wang, Zhiyu Li, and Zhouhan Lin. PonderLM: Pretraining language models to ponder in continuous space. In C. Vondrick, B. Hariharan, C. Raffel, L. Pinto, D. Yang, and A. Faust (eds.), *International Conference on Learning Representations*, volume 2026, pp. 91874–91893, 2026b. URL https://proceedings.iclr.cc/paper_files/paper/2026/file/94572a66c6b0710c5263f15d8ef6b463-Paper-Conference.pdf.

Wenbo Zhang and Xiang Ren. WhiteMatter: All-to-all cross-layer connections via KV mixing, 2026. URL https://arxiv.org/abs/2608.18486.

Xiangdong Zhang, Debing Zhang, Shaofeng Zhang, Xiaohan Qin, Yu Cheng, and Junchi Yan. NITP: Next implicit token prediction for LLM pre-training. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=oPQmCKS1tV.

Zhen Zhang, Xuehai He, Weixiang Yan, Ao Shen, Chenyang Zhao, and Xin Wang. Soft Thinking: Unlocking the reasoning potential of LLMs in continuous concept space. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pp. 168990–169012. Curran Associates, Inc., 2025. doi: 10.52202/085713-5629. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/f7396d1c54d51416958d63e285377103-Paper-Conference.pdf.

Wanjun Zhong, Ruixiang Cui, Yiduo Guo, Yaobo Liang, Shuai Lu, Yanlin Wang, Amin Saied, Weizhu Chen, and Nan Duan. AGIEval: A human-centric benchmark for evaluating foundation models. In Kevin Duh, Helena Gomez, and Steven Bethard (eds.), *Findings of the Association for Computational Linguistics: NAACL 2024*, pp. 2299–2314, Mexico City, Mexico, June 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-naacl.149. URL https://aclanthology.org/2024.findings-naacl.149/.

Hanlin Zhu, Shibo Hao, Zhiting Hu, Jiantao Jiao, Stuart J Russell, and Yuandong Tian. Reasoning by superposition: A theoretical perspective on chain of continuous thought. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pp. 79931–79963. Curran Associates, Inc., 2025. doi: 10.52202/085713-2670. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/72c363c2a573ca2128bd176d3317696b-Paper-Conference.pdf.

Yufan Zhuang, Liyuan Liu, Chandan Singh, Jingbo Shang, and Jianfeng Gao. Mixture of Inputs: Text generation beyond discrete token sampling. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pp. 26363–26387. Curran Associates, Inc., 2025. doi: 10.52202/085713-0889. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/25c61012c845908bff2ea6223cdc6192-Paper-Conference.pdf.

A Additional Method Details

A.1 Prefix State Dropout

For each training sequence, with probability p, we draw a prefix length $m \sim \mathcal{U}\{0, \ldots, n\}$ and feed the fusion layer (Eq. 5)

$$\tilde{\mathbf{e}}_i = \begin{cases} \mathbf{b} & 1 < i \le m \\ E^\top \mathbf{s}_i^\star & i > m \end{cases} \tag{8}$$

in place of the projected state $E^\top \mathbf{s}_i^\star$. The first position always keeps the learned initial-state vector, since no previous output exists there at inference either. Because m ranges up to the full length,

the model also learns to process an entire sequence without states, the mode evaluated as LIFT w/o states (§5.2). The same augmentation is applied to the loss-carrying pass of the adaptation phase (§A.3).

A.2 The State-Alignment Loss

The alignment term $\mathcal{D}_k$ of Eq. 6 compares the teacher's and the student's distributions at the state temperature τ on the k tokens of $\mathbf{s}^\star_{i+1}$, aggregating the remaining probability mass into a single tail outcome. Formally, let S be the k tokens of $\mathbf{s}^\star_{i+1}$, let $\mathbf{p}^\star_\tau = \mathrm{softmax}(\mathbf{o}^\star_i/\tau)$ and $\mathbf{p}_\tau = \mathrm{softmax}(\mathbf{o}_i/\tau)$ be the teacher's and the student's full-vocabulary distributions at temperature τ, and let P and Q be the mass each puts on S. Since the state renormalizes the teacher's probabilities on S (Eq. 4), $p^\star_\tau(v) = P\, s^\star_{i+1}(v)$ for $v \in S$. Then

$$\mathcal{D}_k\big(\mathbf{o}^\star_i \,\|\, \mathbf{o}_i\big) = \tau^2\Big[\sum_{v\in S} P\, s^\star_{i+1}(v) \log \frac{P\, s^\star_{i+1}(v)}{p_\tau(v)} \;+\; (1-P)\log\frac{1-P}{1-Q}\Big], \tag{9}$$

where τ^2 is the usual distillation scaling (Hinton et al., 2015). Keeping only the terms on S would push the student to lower the probability of every token outside S, a constant pressure to sharpen its distribution. With the tail, the loss is minimized exactly when the student matches the teacher on S and puts mass $1-P$ outside it; and as a KL between coarsened distributions, it is a lower bound on the full-vocabulary KL that does not decrease as k grows.

A.3 Adaptation Phase

In the last 10% of training, each microbatch draws its number of passes $R \in \{2, 3\}$ with equal probability and runs:

1. **Pass 1** (no gradient). Every position after the first is fed the bias $\mathbf{b}$, which carries no information about the context; the first position keeps its learned initial state (§3.1). The logits at every position give a state (Eq. 4).
2. **Pass 2** (no gradient, only when $R = 3$). Every position after the first is fed the state that pass 1 predicted at the preceding position, over the whole sequence, yielding refined states.
3. **Final pass** (with gradient). Each sequence independently draws, with probability p, a prefix length $m \sim \mathcal{U}\{0, \ldots, n\}$ (Eq. 8). Positions $1 < i \le m$ are fed $\mathbf{b}$, and all later positions the states of the previous pass. This is the only pass that carries a loss: the language-modeling term of Eq. 6 over every position, with the alignment term off ($\lambda = 0$).

The fed-back states are detached, so no gradient flows between passes, and the no-gradient passes add 1.5 forward passes per step on average (Table 17). The teacher is not needed in this phase.

B Experimental Setup

B.1 S_5 State Tracking: Setup

Models and training. All models of §4 use the OLMo 2 block (§B.2), slightly adapted to their tiny size and small vocabulary: two layers, width 256, four attention heads of dimension 64, a SwiGLU MLP of hidden size 512, RoPE with $\theta = 10{,}000$, and untied input and output embeddings over a synthetic vocabulary of $1{,}047$ tokens (padded to $1{,}152$) that includes the 120 elements of S_5. They are trained in fp32 with AdamW ($\beta = (0.9, 0.95)$, $\epsilon = 10^{-8}$, weight decay 0.1 on weight matrices), a peak learning rate of 10^{-3} with 200 warm-up steps and a cosine decay to $0.01\times$ the peak, gradient clipping at 1.0, and batches of 512 sequences. Each step draws a fresh batch of a single length, chosen uniformly from the training lengths $N \in \{1, 2, 3, 4, 6, 8, 12, 16\}$. The initial state is uniform over S_5, and each a_i is drawn uniformly from 12 fixed generators, which include a transposition and a 5-cycle and thus generate S_5. The input is $\langle \texttt{BOS}, s_0, a_1, \ldots, a_N, = \rangle$, and the answer is predicted at the last position.

LIFT on S_5. These models differ from the recipe of §3 in several respects. The fusion layer is a single linear map, $\hat{\mathbf{h}}^0_i = W_f\,[\mathbf{h}^0_i\,;\, W_e^\top \mathbf{s}_i]$ with $W_f \in \mathbb{R}^{d\times 2d}$, in place of Eq. 5, and the state keeps

the top $k = 256$ tokens at $\tau = 1$. LIFT-RNN is trained on the answer loss alone, backpropagating through all positions. The teacher-fed models add the forward KL to the teacher's full output distribution at every position before the answer, with weight 1, and use no prefix state dropout. The teacher's states are built from its logits, computed in one parallel pass for the Transformer$\times$8 teacher and sequentially, as at inference, for the LIFT-RNN teacher. In the last 10% of steps, the teacher-fed models are instead fed their own states, computed sequentially with no gradient through the states, and the losses are unchanged.

Evaluation. We report the accuracy of the argmax prediction at the answer position. LIFT models are evaluated sequentially, as at inference, each position fed the state predicted at the preceding one; the Transformer answers in one parallel pass. Each length has one evaluation set, shared by all models and seeds and excluded from training: $2{,}000$ sequences, except at $N{=}1$ (200) and $N{=}2$ ($1{,}728$), which have only $1{,}440$ and $17{,}280$ distinct sequences. All models are trained with three seeds (0, 1 and 2), and each teacher-fed LIFT model uses the teacher of its own seed, giving three distinct student–teacher pairs.

B.2 Implementation Details

Training budgets and teachers. The three-scale experiments use training budgets of $5\times$ the Chinchilla token budget (Hoffmann et al., 2022): 13.4B, 34.9B, and 107.4B tokens for the 135M, 350M, and 1B models, respectively (rounded in Table 3). At each scale, we use a pretrained vanilla baseline of the same size as one teacher and a larger pretrained model as the other. The larger teachers for the 135M and 350M models are pretrained vanilla baselines; for the 1B model, we use pretrained OLMo 2-7B (Walsh et al., 2025). The compute-matched budgets are given in §F.1.

Model configurations. All three models use the OLMo 2 block (Walsh et al., 2025): RoPE ($\theta = 500{,}000$), RMSNorm with QK-norm, a SwiGLU MLP (Shazeer, 2020) with ratio 8, a head dimension of 128, and the dolma2 tokenizer (vocabulary $100{,}278$, padded to $100{,}352$). The 1B model is OLMo 2's 1B configuration unchanged; the 135M and 350M models scale down its width and depth, and tie the input and output embeddings. Table 3 lists the configurations; model names refer to non-embedding parameter counts.

Table 3: Model and training configurations. Parameter counts exclude the RMSNorm vectors; the 1B counts both embedding matrices, the smaller models one tied matrix.

	135M	350M	1B
Non-embedding parameters	134.3M	348.9M	1,073.7M
Total parameters	237.0M	490.2M	1,484.9M
Hidden dimension d	1024	1408	2048
Layers	8	11	16
Attention heads	8	11	16
Embeddings	tied	tied	untied
Context length	1024	2048	4096
Global batch (sequences)	1040	512	512
Tokens per step	1.06M	1.05M	2.10M
Training budget ($5\times$ Chinchilla)	13.4B	34.9B	107B
Warm-up (steps / tokens)	2000 / 2.1B	3000 / 3.1B	4000 / 8.4B
Peak learning rate	6×10^{-4}	5×10^{-4}	4×10^{-4}

Optimization. We use AdamW with $\beta = (0.9, 0.95)$, $\epsilon = 10^{-8}$, weight decay 0.1 (not applied to the embeddings), gradient clipping at a norm of 1.0, and a z-loss with weight 10^{-5}, all as in OLMo 2. Training runs in bf16 mixed precision with the cross-entropy loss computed in fp32. The state input, the teacher's top-k distribution, is quantized to 8 bits. Within each model size, LIFT and its vanilla control share the configuration, the random seed and therefore the exact data order.

LIFT hyperparameters. The three-scale experiments use the same LIFT settings at every scale. The state keeps the top $k = 1024$ tokens at temperature $\tau = 1.5$ (Eq. 4); the alignment term uses the same k and τ, with weight $\lambda = 1.5$ (Eq. 6). Prefix state dropout is applied to each training

sequence with probability $p = 0.5$ (§A.1). The adaptation phase covers the last 10% of training, coinciding with the learning-rate anneal, and draws one or two no-gradient passes per step with equal probability (§A.3). The comparison of §5.3 uses its own settings (§B.3).

Pretraining data. The training corpus is sampled from the OLMo 2 stage-1 pretraining mix: DCLM-baseline, StarCoder, peS2o, Proof-Pile-2 and OLMo-Mix, tokenized with the dolma2 tokenizer. Our stratified versions sample whole files per source domain so that the source mixture is preserved: approximately 93.8% DCLM, 2.6% StarCoder, 2.0% peS2o, 1.3% Proof-Pile-2 and 0.4% OLMo-Mix. We use a 400B-token selection for the 350M and 1B runs and a 40B-token selection for the 135M runs; every run consumes its budget from a single pass over a random order of its selection.

Learning-rate schedule. All models follow the WSD protocol of Wen et al. (2025). The learning rate warms up linearly from zero over the steps listed in Table 3, is then held constant at the peak value for the first 90% of the training budget ($4.5\times$ Chinchilla), and is annealed over the last 10% ($0.5\times$ Chinchilla), branching from the trunk checkpoint at 90%. The anneal uses the inverse-proportional decay shape of Wen et al. (2025), in which the reciprocal of the learning rate rises linearly, and ends at $0.1\times$ the peak value. All results reported in §5 are after an identical annealing over the last 10% of training to an LR of $0.1\times$ the starting value.

B.3 Feedback-Transformer Comparison: Setup

Setup. All models of §5.3 are trained with the T^{2}MLR authors' training code on the SmolLM2-135M backbone (Allal et al., 2025): 30 layers, width 576, 9 attention heads with 3 key-value heads, tied embeddings, 135M parameters including embeddings, and the SmolLM2 tokenizer of 49,152 tokens. The data is the 10B-token FineWeb-Edu sample (`sample-10BT`; Penedo et al., 2024) packed into sequences of 2,048 tokens; one epoch is 19,073 steps of 256 sequences. The training configuration is identical for all models: AdamW with $\beta = (0.9, 0.98)$ and weight decay 0.01, a peak learning rate of 5×10^{-4} with a warm-up of 954 steps (5%), a global batch of 256 sequences (0.5M tokens per step), gradient clipping at 1.0, bf16 precision, and a WSD schedule with a linear decay to $0.1\times$ the peak over the last 10% of steps; for LIFT that last 10% is the adaptation phase of §3.2. T^{2}MLR runs in the authors' configuration, T^{2}MLR(13,18) with 16 forward and 4 backward refinement passes, and is evaluated with the exact sequential recurrence, as in their paper. LIFT uses $k = 1024$ and is trained without the KL objective, so that the measured gain comes from state propagation and not from the distillation signal; its teacher is the last checkpoint of the vanilla baseline model (the same model, trained on 23B tokens). The compute-matched budget is T^{2}MLR's 19,073 steps at $2.28\times$ the FLOPs of a vanilla step per token; at $1.36\times$ per token in the trunk and $1.87\times$ in the adaptation phase, LIFT reaches it after 16.2B tokens and the vanilla model after 23.0B. The checkpoint the authors published was trained for 19,294 steps against our 19,073 (1.2% more tokens), with a cosine decay to $0.001\times$ the peak. Parameter counts are 138.2M for LIFT, 136.2M for T^{2}MLR, 135.2M for the Multi-pass Transformer and 134.5M for the vanilla model. Perplexity is measured in fp32 on 127K held-out FineWeb-Edu tokens; the arithmetic and LM-eval protocols are those of §B.4.

Multi-pass Transformer. We take from the paper (Wang et al., 2026) its feedback path, a gated linear unit that fuses the previous top-layer state (the value) with the current token embedding (the gate); its pass schedule, 75% of the batches with a single pass, 22% with two and 3% with three, which is the schedule the authors report for their larger runs; its prefix mixin, which leaves a random prefix of each sequence on plain embeddings in the extra passes, so that training matches the prompt-then-generation structure of inference; and its stabilizers, noise on the carried state and normalization of the fused input (the third, tying the input and output embeddings, the shared backbone already does). We could not take their optimizer, since all models here share one and changing it for a single model would confound the comparison, nor their depth scaling of the residual branches, which alters the backbone shared by all models here and which we replace by carrying the normalized top-layer state. We use the shared 135M, 10B-token setup instead of their 1B model and 400B-token dataset. As with T^{2}MLR, we report the model in its recurrent mode.

B.4 Evaluation Benchmarks

Downstream suite. The twenty suites are run with OLMES (Gu et al., 2025) and its task configurations (Table 4); LIFT scores every prompt and answer with its own states. The window is each model's trained context (1,024 / 2,048 / 4,096 tokens at 135M / 350M / 1B; 2,048 for the SmolLM2 models), and at 135M, generation is capped at 256 tokens, the same for every 135M model, so its generative numbers compare within the size, not across sizes. Cloze scoring appends each answer option to the question and scores it by its length-normalized log-likelihood; the prediction is the highest-scoring option. Bits per byte is $-\log_2 P(\text{gold answer})$ divided by the answer's UTF-8 bytes. The main table keeps a suite if, at 1B, at least one model is five points above chance (multiple choice) or above 5% (generative); bits-per-byte suites always enter. This leaves out the letter-format multiple choice (MMLU, AGI-Eval), where every model is at chance, and generative math and code (GSM8K, MATH, HumanEval, MBPP), where every model is under 3%. BoolQ is left out as well: a yes/no cloze whose majority class is 62%, which several models fall below and which moves by ten points across training seeds. Every excluded suite is reported in Tables 7–15.

Arithmetic. Expressions are generated by drawing n integers uniformly from 0 to 9 and $n-1$ operators uniformly from $\{+,-\}$, and are evaluated left to right; duplicates are discarded, which is why the $n=2$ bin holds all 200 possible expressions. The answer is an integer, possibly negative. Every prompt consists of the same three worked examples (3-shot), followed by the query, one expression per line, and the scored continuation is the answer with its leading space:

```
4 - 6 = -2
0 + 5 + 7 - 1 - 3 = 8
3 - 3 - 1 = -1
2 + 0 - 6 + 7 + 5 =
```

with answer `8`. Bits per byte is $-\log_2 P(\text{answer})$ summed over the set and divided by the total number of UTF-8 bytes of the scored answers. The exact-match accuracy of the teacher-forced argmax is recorded alongside it (Table 12): it is substantial only at two operands and a few percent from three operands on, for every model, which is why Table 1 reports bits per byte.

Table 4: The downstream suite. Items: scored instances (per task for the core tasks); Shots: in-context examples; Group: the Table 1 column, or the reason a suite is reported only in the appendix. Task sources are cited below the table.

Suite	Tests	Items	Shots	Scoring	Group
Core 8: ARC-Easy, ARC-Challenge, CommonsenseQA, HellaSwag, OpenBookQA, PIQA, SocialIQA, WinoGrande	Science and commonsense questions	500–1,267	5	cloze acc.	MC
MMLU	57 academic subjects	14,042	5	cloze acc.	MC
Basic skills (OLMES)	Arithmetic, string operations, pattern continuation, coding, logical reasoning, common knowledge	5,967	5	cloze acc.	MC
Generative QA: CoQA, SQuAD, Jeopardy, Natural Questions, DROP	Conversational, extractive and closed-book QA	7,983 / 1,000 / 2,116 / 1,000 / 1,000	5 (CoQA 0)	F1	Gen.
TriviaQA	Closed-book trivia	7,993	5	F1	Gen.
BBH	27 reasoning tasks with chain of thought	6,511	3	exact match	Gen.
ARC, MMLU, basic skills; MATH; HumanEval; MBPP	The gold answer's likelihood	3,548 / 14,042 / 5,967 / 5,000 / 164 / 500	5 / 5 / 5 / 4 / 3 / 3	bits per byte	RC
BoolQ	Yes/no questions on a passage	1,000	5	cloze acc.	coin flip
MMLU, letter format	As above, scored by the option letter	14,042	5	letter acc.	at chance
AGI-Eval English	Exam questions, letter format	2,646	1	letter acc.	at chance
GSM8K	Grade-school math, chain of thought	1,319	5 and 8	exact match	under 3%
MATH-500, Minerva MATH	Competition math	500, 5,000	4	exact match	under 3%
HumanEval, MBPP	Program synthesis, 20 samples at T=0.8	164, 500	3	pass@1	under 3%

Sources: ARC (Clark et al., 2018), CommonsenseQA (Talmor et al., 2019), HellaSwag (Zellers et al., 2019), OpenBookQA (Mihaylov et al., 2018), PIQA (Bisk et al., 2020), SocialIQA (Sap et al., 2019), WinoGrande (Sakaguchi et al., 2021), MMLU (Hendrycks et al., 2021a), CoQA (Reddy et al., 2019), SQuAD (Rajpurkar et al., 2016), Natural Questions (Kwiatkowski et al., 2019), DROP (Dua et al., 2019), TriviaQA (Joshi et al., 2017), BBH (Suzgun et al., 2023), MATH (Hendrycks et al., 2021b), HumanEval (Chen et al., 2021), MBPP (Austin et al., 2021), BoolQ (Clark et al., 2019), AGI-Eval (Zhong et al., 2024), GSM8K (Cobbe et al., 2021).

C Additional Results

C.1 Transformer Training Budget on S_5

§4 attributes the two-layer Transformer's ceiling on S_5 to depth. Every S_5 model has $d_{\text{model}} = 256$ and trains with batch size 512 for 5,000 steps unless stated otherwise. A competing explanation is that the model is simply undertrained, so we trained the same baseline at four budgets, all else held fixed: the token-matched 5,000 steps, $8\times$ (40,000), $20\times$ (100,000) and $40\times$ (200,000). Figure 4 reports all four, with three seeds each, under the protocol of §4.

The model gains nothing after $20\times$, where the $20\times$ and $40\times$ curves coincide, and it remains at $\approx 10\%$ at N=12 and at chance at N=16 and every longer length, in every seed of every budget.

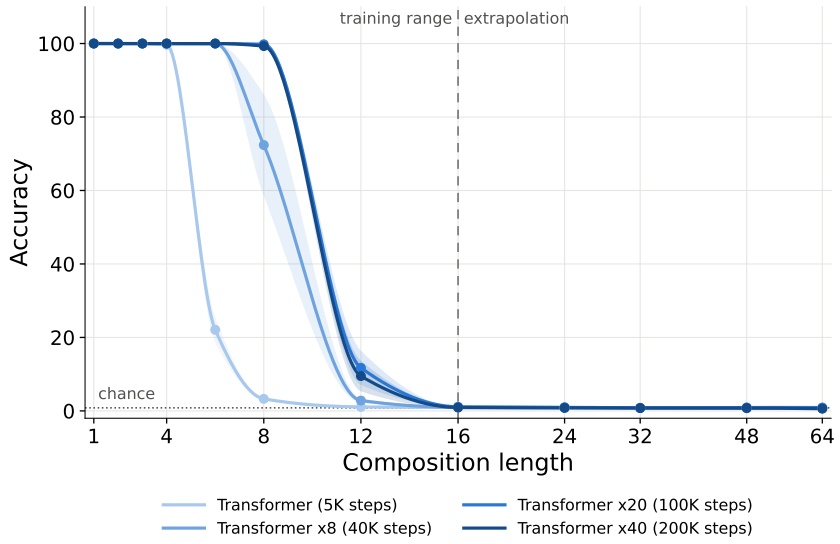


Figure 4: The same two-layer Transformer on S_5 at four training budgets, darker lines for longer training. Bands show standard error over three seeds. Training beyond 100K steps ($20\times$) brings no further gain.

C.2 Perplexity by Source and Evaluation Uncertainty

Table 5 reports perplexity on every source of the OLMo 2 perplexity evaluation set: the full C4 validation subset and 10% of each other source. LIFT is ahead of the token-matched, compute-matched and distillation baselines on all eleven sources at every size. Table 6 gives, for the paper's training seed, a paired bootstrap interval over the evaluation sequences for every perplexity gap of Table 1; it measures how much a gap depends on the held-out text at a fixed training run, not seed variance.

C.3 Downstream Results per Suite

Tables 7–9 give every OLMES suite for the rows of Table 1 and LIFT with the other teachers of §B.2, including the suites left out of the main table.

C.4 Seed Variance at 135M

Table 10 gives the 135M rows of Table 1 over three training seeds.

C.5 Arithmetic by Number of Operands

Tables 11 and 12 break the arithmetic results down by the number of operands: bits per byte of the answer, and exact-match accuracy of the teacher-forced argmax.

C.6 Pattern Continuation

Table 13 gives the pattern-continuation results discussed in §5.2.

C.7 Feedback-Transformer Comparison: Results

Seeds. Every model of Table 2 except T^{2}MLR (HF) is trained with three seeds (42, 43, 44). Table 14 reports the mean and standard deviation of each column; Table 2 shows the means.

Table 15 gives every OLMES suite for §5.3 (seed 42; three-seed statistics in Table 14).

Table 5: Perplexity by source on the OLMo 2 perplexity evaluation set: the full C4 validation slice (971 sequences of 1,024 tokens) and 10% of each other source (at least 50 sequences). Columns as in Table 1; lower is better, bold is the best in a row.

Source	LIFT	w/o states	Tr. token	Tr. compute	Distillation	Seqs
135M						
C4	**30.65**	32.44	32.43	31.04	31.77	971
Common Crawl	**31.55**	33.44	33.55	32.01	32.63	50
Books	**33.58**	35.93	35.90	33.85	34.86	50
Wikipedia	**19.64**	20.70	20.78	19.87	20.39	50
peS2o	**17.85**	18.93	19.00	18.12	18.56	51
S2ORC	**38.88**	40.91	41.50	39.69	40.34	97
Reddit	**33.89**	35.83	35.74	34.37	34.88	50
Stack (code)	**7.10**	7.45	7.59	7.20	7.38	50
Pile	**18.20**	19.29	19.30	18.52	18.87	66
ICE	**30.52**	32.98	33.92	31.23	32.28	89
WikiText-103	**24.69**	26.72	26.88	25.23	26.30	50
350M						
C4	**23.82**	24.96	25.15	24.32	24.52	971
Common Crawl	**24.70**	25.90	26.02	25.06	25.32	50
Books	**24.29**	25.69	25.94	24.76	25.12	50
Wikipedia	**15.85**	16.52	16.75	16.11	16.29	50
peS2o	**12.48**	12.99	13.04	12.67	12.77	51
S2ORC	**27.44**	28.68	29.32	28.46	28.15	97
Reddit	**26.79**	28.17	28.30	27.49	27.65	50
Stack (code)	**5.94**	6.13	6.25	6.04	6.04	50
Pile	**14.07**	14.77	14.97	14.49	14.64	66
ICE	**24.51**	26.00	26.73	25.61	26.44	89
WikiText-103	**18.43**	19.56	19.59	18.86	19.07	50
1B						
C4	**18.81**	19.62	19.79	19.17	19.40	971
Common Crawl	**19.62**	20.37	20.59	19.96	20.16	50
Books	**17.82**	18.71	19.05	18.27	18.44	50
Wikipedia	**12.31**	12.73	12.88	12.46	12.68	50
peS2o	**10.07**	10.39	10.53	10.26	10.33	51
S2ORC	**22.38**	23.05	23.69	23.12	23.63	97
Reddit	**21.74**	22.63	22.78	22.22	22.34	50
Stack (code)	**4.85**	4.97	5.03	4.89	4.93	50
Pile	**11.28**	11.71	11.87	11.50	11.61	66
ICE	**19.34**	20.23	20.43	19.91	19.96	89
WikiText-103	**13.79**	14.36	14.45	13.96	14.17	50

Table 6: Evaluation-set uncertainty of the perplexity gaps of Table 1, for the paper's training seed: the gap of LIFT to each Transformer baseline with a paired bootstrap 95% interval over the evaluation sequences (both models scored on the same text; 5,000 resamples). This measures how much a gap depends on the held-out text, at a fixed training run; it is not seed variance, which Table 10 reports for 135M. Negative is in favor of LIFT.

Δ PPL	135M	350M	1B
vs. Transformer, token-match	-1.77 [-1.85, -1.70]	-1.33 [-1.39, -1.28]	-0.99 [-1.05, -0.93]
vs. Transformer, compute-match	-0.38 [-0.44, -0.33]	-0.50 [-0.55, -0.46]	-0.36 [-0.40, -0.32]
vs. Distillation	-1.11 [-1.17, -1.06]	-0.70 [-0.74, -0.66]	-0.59 [-0.63, -0.55]

Table 7: OLMES results at 135M for the rows of Table 1 and LIFT with the other teachers of §B.2: the suites of the three Table 1 groups, then the suites left out of it (at chance or floor for every model, and BoolQ). Accuracy % for the multiple-choice, generative and excluded rows; bits per byte (lower is better) for the BPB rows; suites as in Table 4. Gray: the 2×-token teacher, outside the bold rule. Bold: best in the row.

Suite	LIFT 135M@29.6B	w/o states 135M@29.6B	Tr. token 13.4B	Tr. compute 19.3B	Distillation 13.4B	2× tokens 29.6B
Core 8 (cloze)	**44.0**	43.3	42.5	43.9	43.9	44.7
MMLU (cloze)	**28.1**	27.9	27.3	28.0	27.6	28.2
Basic skills (cloze)	46.0	44.8	44.6	**46.4**	45.8	47.2
Generative QA	13.4	11.3	11.4	**13.8**	10.7	15.5
TriviaQA	13.6	13.3	13.8	**14.8**	14.3	14.8
BBH (CoT)	**19.5**	11.3	15.6	13.8	12.9	14.7
ARC BPB	1.039	1.076	1.082	**1.032**	1.039	0.990
MMLU BPB	**1.298**	1.336	1.347	1.314	1.313	1.276
Basic skills BPB	1.338	1.367	1.377	**1.327**	1.340	1.308
MATH BPB	**0.998**	1.024	1.022	1.000	1.013	0.976
HumanEval BPB	**0.842**	0.866	0.853	**0.842**	0.871	0.843
MBPP BPB	**0.951**	0.965	0.976	0.967	0.960	0.947
Not in Table 1						
BoolQ (cloze)	63.8	63.8	59.9	58.8	**63.9**	63.3
MMLU (letter)	25.3	**26.1**	**26.1**	**26.1**	25.7	24.9
AGI-Eval (letter)	21.6	21.9	22.3	21.6	**23.6**	23.0
GSM8K (5-shot)	**2.3**	**2.3**	1.4	1.7	1.8	2.3
GSM8K (8-shot)	2.2	2.0	**2.7**	1.9	2.5	2.6
MATH-500	**0.2**	0.0	**0.2**	0.0	**0.2**	0.2
Minerva MATH	**0.2**	0.0	0.0	0.0	0.1	0.2
HumanEval pass@1	**0.2**	0.1	0.1	0.1	0.0	0.2
MBPP pass@1	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	0.0

Table 8: OLMES results at 350M for the rows of Table 1 and LIFT with the other teachers of §B.2: the suites of the three Table 1 groups, then the suites left out of it (at chance or floor for every model, and BoolQ). Accuracy % for the multiple-choice, generative and excluded rows; bits per byte (lower is better) for the BPB rows; suites as in Table 4. Gray: the 2×-token teacher, outside the bold rule. Bold: best in the row.

Suite	LIFT 350M@77.8B	w/o states 350M@77.8B	Tr. token 34.9B	Tr. compute 49.9B	Distillation 34.9B	2× tokens 77.8B
Core 8 (cloze)	**51.2**	49.8	48.9	50.8	50.4	51.7
MMLU (cloze)	30.2	30.1	30.3	**31.0**	30.6	31.1
Basic skills (cloze)	**53.8**	52.7	50.3	50.4	52.4	52.6
Generative QA	**27.3**	24.0	21.5	23.1	24.8	26.6
TriviaQA	**21.9**	20.6	20.1	21.7	21.4	23.6
BBH (CoT)	20.3	18.4	**21.3**	**21.3**	20.1	21.9
ARC BPB	**0.872**	0.893	0.905	0.893	0.875	0.865
MMLU BPB	**1.139**	1.164	1.173	1.154	1.155	1.129
Basic skills BPB	**1.099**	1.136	1.169	1.153	1.129	1.153
MATH BPB	**0.859**	0.878	0.886	0.865	0.872	0.849
HumanEval BPB	0.735	0.758	0.751	0.750	**0.729**	0.694
MBPP BPB	**0.825**	0.840	0.851	0.839	0.832	0.815
Not in Table 1						
BoolQ (cloze)	**63.7**	63.6	61.7	55.9	46.2	58.1
MMLU (letter)	24.3	24.3	25.0	**26.9**	25.0	25.9
AGI-Eval (letter)	22.7	22.2	**23.0**	22.2	21.4	23.8
GSM8K (5-shot)	1.4	1.4	**2.0**	**2.0**	1.4	2.7
GSM8K (8-shot)	2.3	1.9	**2.4**	1.8	1.7	3.0
MATH-500	1.0	**1.2**	0.0	0.2	0.4	1.0
Minerva MATH	**0.9**	0.6	0.2	0.2	0.5	0.8
HumanEval pass@1	0.2	0.2	0.3	**0.4**	0.2	0.5
MBPP pass@1	0.1	**0.2**	0.1	**0.2**	0.1	0.3

Table 9: OLMES results at 1B for the rows of Table 1 and LIFT with the other teachers of §B.2: the suites of the three Table 1 groups, then the suites left out of it (at chance or floor for every model, and BoolQ). Accuracy % for the multiple-choice, generative and excluded rows; bits per byte (lower is better) for the BPB rows; suites as in Table 4. Gray: the 2×-token teacher, outside the bold rule. Bold: best in the row.

Suite	LIFT 1B@222B	LIFT 1B@4001B	w/o states 1B@222B	w/o states 1B@4001B	Tr. token 107.4B	Tr. compute 152.4B	Distillation 107.4B	2× tokens 222B
Core 8 (cloze)	**59.4**	58.3	58.3	56.7	58.0	59.0	58.5	60.2
MMLU (cloze)	**35.4**	35.1	34.7	34.3	34.4	34.9	35.0	35.6
Basic skills (cloze)	**62.6**	62.2	60.6	61.5	59.3	60.9	62.5	63.0
Generative QA	**42.2**	42.1	39.3	39.5	38.6	40.5	41.4	43.5
TriviaQA	37.3	37.6	36.1	35.7	34.9	**37.9**	36.7	41.2
BBH (CoT)	25.6	**27.8**	23.2	26.2	25.2	26.0	26.4	27.4
ARC BPB	0.753	**0.752**	0.768	0.774	0.770	0.771	0.759	0.744
MMLU BPB	**0.991**	**0.991**	1.008	1.011	1.023	1.004	1.003	0.979
Basic skills BPB	**0.845**	0.866	0.885	0.874	0.955	0.919	0.884	0.858
MATH BPB	0.734	**0.731**	0.750	0.748	0.762	0.742	0.740	0.726
HumanEval BPB	0.596	0.579	0.611	**0.577**	0.599	0.589	0.584	0.565
MBPP BPB	**0.711**	0.716	0.724	0.733	0.740	0.728	0.717	0.711
Not in Table 1								
BoolQ (cloze)	64.9	60.4	65.7	53.2	63.8	64.2	**66.8**	55.4
MMLU (letter)	24.8	23.3	25.0	23.9	26.4	**27.1**	25.0	28.9
AGI-Eval (letter)	21.7	21.7	20.6	20.7	**24.6**	21.7	21.4	23.1
GSM8K (5-shot)	2.0	2.6	2.0	1.9	2.9	2.7	**3.0**	2.4
GSM8K (8-shot)	**2.5**	2.3	2.4	2.4	2.0	2.3	2.0	1.8
MATH-500	**2.6**	1.6	0.4	0.0	1.6	0.6	1.2	0.4
Minerva MATH	**1.2**	0.7	0.5	0.3	1.0	1.0	0.7	0.9
HumanEval pass@1	**2.7**	2.5	2.3	1.9	1.7	2.1	2.0	2.3
MBPP pass@1	**1.0**	0.8	0.7	0.5	0.4	0.7	**1.0**	1.0

Table 10: The 135M rows of Table 1 over three training seeds (6198 / 6199 / 6200): mean ± sd of each column. We regard a difference between two models as beyond seed noise when their mean ± sd ranges do not overlap. Pattern: the basic-skills pattern-continuation task (534 questions).

Model	PPL ↓	MC %	Gen. %	RC (BPB ↓)	Basic skills ↓	Pattern %	Pattern BPB ↓
LIFT	30.57 ± 0.08	39.5 ± 0.1	15.0 ± 0.6	1.076 ± 0.006	1.326 ± 0.018	53.2 ± 0.7	1.339 ± 0.045
LIFT w/o states	32.39 ± 0.07	38.8 ± 0.2	11.9 ± 0.5	1.105 ± 0.005	1.364 ± 0.007	53.2 ± 0.7	1.380 ± 0.046
Transformer, token-match	32.44 ± 0.02	38.4 ± 0.5	13.0 ± 0.6	1.112 ± 0.004	1.390 ± 0.011	50.7 ± 2.0	1.427 ± 0.051
Transformer, compute-match	31.04 ± 0.00	39.2 ± 0.3	13.6 ± 0.6	1.085 ± 0.005	1.348 ± 0.036	52.2 ± 0.5	1.393 ± 0.003
Distillation	31.81 ± 0.04	39.2 ± 0.4	13.0 ± 0.4	1.096 ± 0.006	1.361 ± 0.019	50.5 ± 1.4	1.373 ± 0.022

Table 11: Arithmetic by number of operands: bits per byte of the answer (lower is better), for the rows of Table 1 and LIFT with the other teachers of §B.2 plus the vanilla Transformer trained on 2× the tokens (Figure 3a); 1,000 expressions per operand count (200 at two operands), three-shot. Avg. is over all expressions (total bits over total answer bytes), the number reported in Table 1. Bold: best in the column within a size.

	Model	Tokens	Teacher (size@tokens)	Number of operands n 2	3	4	5	6	7	8	9	10	Avg.
135M	LIFT	13.4B	135M@29.6B	2.03	**2.13**	**2.12**	2.27	2.27	2.35	2.36	2.42	2.42	**2.29**
	LIFT	13.4B	350M@38.4B	2.23	2.21	2.17	**2.26**	**2.25**	**2.32**	**2.35**	**2.38**	**2.41**	**2.29**
	LIFT w/o states	13.4B	135M@29.6B	2.24	2.32	2.26	2.37	2.37	2.47	2.49	2.55	2.56	2.42
	LIFT w/o states	13.4B	350M@38.4B	2.33	2.28	2.24	2.44	2.45	2.58	2.59	2.63	2.64	2.48
	Transformer, token-match	13.4B	–	**2.02**	2.28	2.29	2.42	2.45	2.55	2.57	2.58	2.59	2.46
	Transformer, compute-match	19.3B	–	2.03	2.21	2.32	2.48	2.52	2.61	2.62	2.62	2.65	2.50
	Transformer, 2× tokens	29.6B	–	2.20	2.28	2.29	2.37	2.37	2.44	2.45	2.47	2.50	2.39
	Distillation	13.4B	135M@29.6B	2.03	2.18	2.22	2.32	2.33	2.43	2.45	2.48	2.49	2.36
350M	LIFT	34.9B	350M@77.8B	1.56	1.97	2.00	**2.11**	**2.13**	**2.18**	**2.19**	**2.24**	**2.24**	**2.12**
	LIFT	34.9B	1B@111.8B	**1.50**	**1.94**	**1.99**	**2.11**	2.16	**2.18**	2.22	2.27	2.28	2.14
	LIFT w/o states	34.9B	350M@77.8B	1.64	2.00	2.07	2.20	2.21	2.28	2.30	2.35	2.36	2.21
	LIFT w/o states	34.9B	1B@111.8B	1.55	2.10	2.16	2.30	2.33	2.36	2.39	2.43	2.44	2.30
	Transformer, token-match	34.9B	–	2.04	2.27	2.24	2.38	2.45	2.54	2.59	2.60	2.62	2.46
	Transformer, compute-match	49.9B	–	1.84	2.06	2.11	2.25	2.33	2.42	2.49	2.49	2.51	2.33
	Transformer, 2× tokens	77.8B	–	1.64	2.00	2.06	2.15	2.24	2.30	2.38	2.40	2.44	2.24
	Distillation	34.9B	350M@77.8B	1.83	2.07	2.08	2.20	2.20	2.27	2.31	2.36	2.41	2.23
1B	LIFT	107.4B	1B@222B	0.74	1.86	1.93	1.96	1.99	2.04	2.07	2.11	2.13	1.99
	LIFT	107.4B	1B@4001B	**0.71**	1.84	1.91	1.98	2.02	2.07	2.12	2.18	2.21	2.02
	LIFT	107.4B	7B@701B	0.79	1.79	**1.83**	**1.88**	**1.92**	**1.98**	**2.02**	**2.05**	**2.06**	**1.92**
	LIFT w/o states	107.4B	1B@222B	0.93	1.98	1.98	2.01	2.03	2.08	2.12	2.18	2.19	2.05
	LIFT w/o states	107.4B	1B@4001B	0.77	1.91	1.94	1.98	1.99	2.03	2.07	2.11	2.12	1.99
	LIFT w/o states	107.4B	7B@701B	1.04	1.87	1.88	1.93	1.96	2.01	2.05	2.08	2.10	1.97
	Transformer, token-match	107.4B	–	1.04	1.86	1.97	2.01	2.03	2.07	2.12	2.15	2.16	2.03
	Transformer, compute-match	152.4B	–	0.95	1.94	2.01	2.02	2.05	2.11	2.17	2.20	2.21	2.07
	Transformer, 2× tokens	222B	–	0.78	1.85	1.94	1.96	2.03	2.06	2.10	2.15	2.15	2.01
	Distillation	107.4B	1B@222B	0.77	**1.73**	1.92	1.93	1.95	2.00	2.05	2.07	2.09	1.95

Table 12: Arithmetic by number of operands: exact-match accuracy % of the teacher-forced argmax, for the expressions and rows of Table 11. Avg. is over all expressions. Bold: best in the column within a size.

	Model	Tokens	Teacher (size@tokens)	Number of operands n 2	3	4	5	6	7	8	9	10	Avg.
135M	LIFT	13.4B	135M@29.6B	**7.0**	5.0	3.4	2.4	3.2	2.5	2.2	3.0	1.9	3.0
	LIFT	13.4B	350M@38.4B	5.5	4.4	**4.4**	**3.3**	**4.0**	2.5	2.2	2.5	**3.6**	**3.4**
	LIFT w/o states	13.4B	135M@29.6B	6.0	4.5	3.8	3.1	3.1	2.7	1.7	**3.1**	2.8	3.2
	LIFT w/o states	13.4B	350M@38.4B	5.0	4.7	3.5	**3.3**	3.9	2.0	2.3	2.8	2.3	3.1
	Transformer, token-match	13.4B	–	**7.0**	4.5	4.1	3.0	2.6	**3.0**	2.4	2.0	2.9	3.2
	Transformer, compute-match	19.3B	–	5.5	4.6	3.3	2.2	2.9	2.7	**3.0**	3.0	2.0	3.0
	Transformer, 2× tokens	29.6B	–	4.5	4.6	3.4	2.3	2.8	**3.0**	2.8	2.6	2.8	3.1
	Distillation	13.4B	135M@29.6B	**7.0**	**5.2**	3.9	2.9	2.6	2.2	2.5	2.1	2.4	3.1
350M	LIFT	34.9B	350M@77.8B	21.5	**6.4**	4.2	**4.2**	2.4	2.7	**3.3**	2.5	2.5	**4.0**
	LIFT	34.9B	1B@111.8B	22.5	5.6	**5.3**	4.1	2.5	2.2	3.0	3.1	1.9	3.9
	LIFT w/o states	34.9B	350M@77.8B	15.0	**6.4**	4.3	3.6	3.1	1.9	2.2	3.1	2.6	3.7
	LIFT w/o states	34.9B	1B@111.8B	**23.0**	5.4	4.9	3.8	2.0	2.5	2.5	2.9	2.3	3.8
	Transformer, token-match	34.9B	–	9.5	5.1	4.3	3.4	2.8	3.0	2.2	2.2	**3.1**	3.4
	Transformer, compute-match	49.9B	–	8.5	4.9	4.6	2.9	1.9	**3.3**	2.7	2.5	2.2	3.3
	Transformer, 2× tokens	77.8B	–	13.0	6.1	3.3	3.2	2.9	2.7	2.9	**3.4**	2.7	3.6
	Distillation	34.9B	350M@77.8B	11.0	5.1	4.1	3.8	**3.4**	2.1	2.7	2.9	2.8	3.5
1B	LIFT	107.4B	1B@222B	58.5	8.7	5.3	5.2	**4.9**	3.7	3.0	3.0	2.6	5.9
	LIFT	107.4B	1B@4001B	63.0	12.8	4.4	4.5	4.3	**4.5**	3.4	2.6	2.0	6.2
	LIFT	107.4B	7B@701B	**63.5**	11.4	5.7	4.5	3.7	2.9	**3.7**	3.6	2.5	6.2
	LIFT w/o states	107.4B	1B@222B	46.5	8.9	4.6	4.5	4.1	2.8	2.6	1.9	2.3	5.0
	LIFT w/o states	107.4B	1B@4001B	59.5	10.8	4.4	3.9	4.1	3.6	3.0	3.1	**3.5**	5.9
	LIFT w/o states	107.4B	7B@701B	47.0	10.8	5.3	4.3	2.9	2.7	3.4	**4.5**	3.0	5.6
	Transformer, token-match	107.4B	–	46.5	8.1	4.6	4.1	4.1	2.1	3.3	4.0	2.2	5.1
	Transformer, compute-match	152.4B	–	48.5	9.0	3.8	3.5	3.7	2.0	3.0	3.4	2.2	4.9
	Transformer, 2× tokens	222B	–	56.0	13.3	4.1	**5.4**	3.8	3.0	2.4	3.3	2.3	6.0
	Distillation	107.4B	1B@222B	56.5	**14.1**	**6.2**	5.1	4.8	2.5	2.4	3.4	2.9	**6.4**

Table 13: Pattern continuation, the basic-skills task of OLMES (534 five-shot questions, e.g., `2 4 6 8` → `10`): cloze accuracy % and bits per byte of the answer (BPB, lower is better), for the rows of Table 1 at the paper's training seed, and the Transformer trained on 2× the tokens (gray, outside the bold rule). Three-seed statistics at 135M are in Table 10. Bold: best in column.

	135M		350M		1B	
Model	Acc. %	BPB ↓	Acc. %	BPB ↓	Acc. %	BPB ↓
LIFT	**53.2**	**1.356**	**64.4**	1.134	**72.1**	**0.851**
LIFT w/o states	52.6	1.383	63.7	**1.121**	70.6	0.965
Transformer, token-match	50.7	1.443	58.4	1.216	66.1	0.948
Transformer, compute-match	52.6	1.396	58.1	1.144	67.6	0.955
Distillation	48.9	1.392	60.7	1.153	69.7	0.890
Transformer, 2× tokens	53.6	1.352	63.5	1.180	70.4	0.896

Table 14: The models of Table 2 over three training seeds (42 / 43 / 44): mean ± sd of each column. T^{2}MLR (HF) is not listed: it is the authors' single released checkpoint, so it has no seed statistics. We regard a difference between two models as beyond seed noise when their mean ± sd ranges do not overlap. T^{2}MLR (our run) sets the compute budget, so its row is the same under both budgets. Bold: the best mean within each budget. Per-suite results of seed 42 are in Table 15.

Model	PPL ↓	Arith. ↓	MC %	Gen. %	RC ↓
Token-matched (10B tokens)					
LIFT	**19.42** ± 0.02	**2.182** ± 0.007	**36.7** ± 0.27	**7.8** ± 0.61	**1.705** ± 0.023
Transformer	20.30 ± 0.03	2.345 ± 0.156	36.2 ± 0.19	6.7 ± 0.21	1.761 ± 0.002
T^{2}MLR (our run)	19.92 ± 0.06	2.267 ± 0.101	36.5 ± 0.04	**7.8** ± 0.94	1.737 ± 0.010
Multi-pass Transformer	20.40 ± 0.04	2.249 ± 0.064	36.3 ± 0.36	6.5 ± 0.25	1.773 ± 0.017
Compute-matched (2.3×10^{19} FLOPs)					
LIFT	18.04 ± 0.04	**2.184** ± 0.032	38.0 ± 0.19	**9.5** ± 0.33	1.629 ± 0.016
Transformer	**18.02** ± 0.02	2.296 ± 0.058	**38.1** ± 0.37	8.5 ± 0.23	**1.622** ± 0.008
T^{2}MLR (our run)	19.92 ± 0.06	2.267 ± 0.101	36.5 ± 0.04	7.8 ± 0.94	1.737 ± 0.010
Multi-pass Transformer	18.58 ± 0.03	2.249 ± 0.037	37.6 ± 0.02	7.7 ± 0.17	1.675 ± 0.008

Table 15: OLMES results for the feedback-Transformer comparison (Table 2), every suite of the scan; the T^{2}MLR columns coincide across the two budgets. Rows and units as in Table 7. Bold: best within a budget.

	Token-matched 10B tokens					**Compute-matched** 2.3×10^{19} FLOPs				
Suite	LIFT	Trans-former	T^{2}MLR (ours)	T^{2}MLR (HF)	Multi-pass	LIFT	Trans-former	T^{2}MLR (ours)	T^{2}MLR (HF)	Multi-pass
Core 8 (cloze)	42.0	40.7	**42.5**	40.7	41.9	43.8	**44.1**	42.5	40.7	43.4
MMLU (cloze)	27.4	**27.5**	27.2	26.5	26.9	27.9	**28.3**	27.2	26.5	27.9
Basic skills (cloze)	**40.4**	39.8	39.6	38.7	40.1	42.1	**42.9**	39.6	38.7	41.4
Generative QA	5.2	4.5	**5.7**	3.8	4.6	**8.0**	7.8	5.7	3.8	6.6
TriviaQA	4.1	**4.2**	4.0	**4.2**	3.6	3.9	3.7	4.0	**4.2**	**4.2**
BBH (CoT)	**13.9**	10.8	13.7	11.6	11.7	**17.5**	14.7	13.7	11.6	12.7
ARC BPB	**1.047**	1.078	1.077	1.110	1.067	1.004	**0.993**	1.077	1.110	1.018
MMLU BPB	**1.359**	1.390	1.384	1.442	1.392	**1.319**	1.321	1.384	1.442	1.335
Basic skills BPB	**1.593**	1.680	1.637	1.744	1.671	1.521	**1.515**	1.637	1.744	1.546
MATH BPB	**1.864**	1.944	1.937	2.011	2.015	**1.808**	1.877	1.937	2.011	1.943
HumanEval BPB	**2.121**	2.268	2.161	2.273	2.188	2.050	**2.044**	2.161	2.273	2.093
MBPP BPB	**2.091**	2.215	2.176	2.319	2.201	**1.964**	1.967	2.176	2.319	2.077
Not in Table 2										
BoolQ (cloze)	61.1	56.2	44.1	**63.9**	46.9	63.8	61.6	44.1	**63.9**	58.7
MMLU (letter)	25.3	25.2	26.0	25.2	**27.0**	25.4	25.8	**26.0**	25.2	25.8
AGI-Eval (letter)	21.4	**23.6**	22.3	22.9	22.7	20.7	21.4	22.3	22.9	**23.5**
GSM8K (5-shot)	2.4	**2.6**	1.7	1.4	1.7	**2.6**	2.4	1.7	1.4	2.1
GSM8K (8-shot)	**2.4**	2.0	1.4	1.6	1.7	**2.4**	1.7	1.4	1.6	1.8
MATH-500	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**
Minerva MATH	0.0	0.0	**0.1**	0.0	0.0	0.0	**0.1**	**0.1**	0.0	0.0
HumanEval pass@1	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**
MBPP pass@1	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**	**0.0**

D ABLATIONS

Beyond a Transformer, LIFT spends about one forward pass per training token, the teacher's. The compute-matched Transformer spends the same compute on over 40% more tokens instead, and LIFT is ahead of it at every scale (Table 1). Here we ask where that lead comes from, changing one thing at a time: how the teacher's forward pass is used (§D.1), how wide the fed-back state is (§D.2), and whether the gain is built in the teacher-forced trunk or in the adaptation phase (§D.3). Every variant keeps the architecture, recipe, learning-rate schedule and data order of §B.2. Figures 5 and 6 plot each variant's improvement over the compute-matched Transformer, so the zero line stands for spending the extra pass on more data. We lead with perplexity on the full C4 validation subset, which separates the variants well beyond seed noise, and report the downstream columns of Table 1 after it.

D.1 TEACHER FORCING: INPUTS, TARGETS, OR NO TEACHER

We compare four ways to spend the extra forward pass:

1. **LIFT**: the teacher's states are both the inputs and the alignment target (§3.2).
2. **LIFT, teacher as input only**: LIFT with $\lambda = 0$ throughout, so the teacher's states are inputs and never targets.
3. **LIFT, no teacher (1 Jacobi iteration)**: no teacher at any point; the extra pass is the model's own. Each training step first runs a pass without gradient in which every position after the first receives the bias $\mathbf{b}$, and the states it predicts (Eq. 4) are fed to the pass that carries the loss, with prefix state dropout as usual and $\lambda = 0$. This is the adaptation phase (§A.3) run from the first step, one Jacobi iteration without gradient between the passes. The last 10% of training is LIFT's adaptation phase, unchanged.
4. **Distillation**: the teacher's states are the target only, and nothing is fed back (the baseline of Table 1).

Every variant is trained at 135M with the three seeds of Table 10.

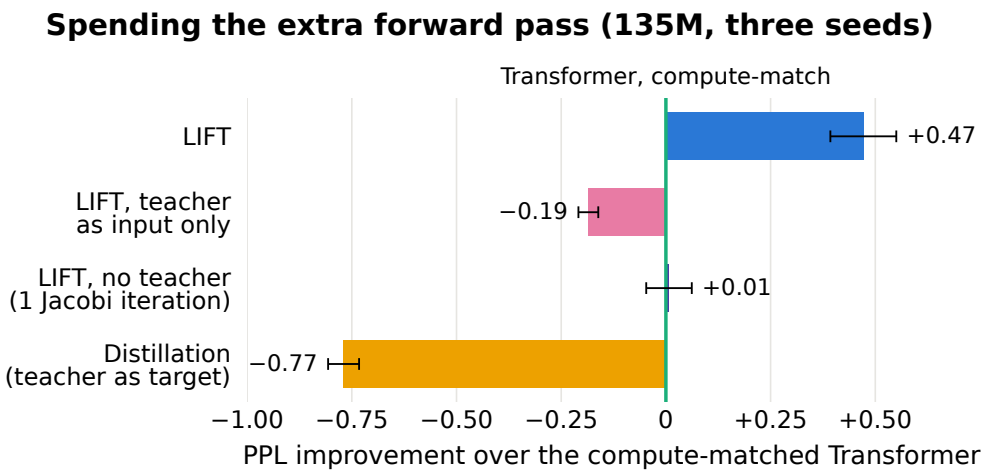


Figure 5: Teacher ablation at 135M: spending LIFT's extra forward pass. Each bar is a model's perplexity improvement on the full C4 validation subset over the compute-matched Transformer, which spends that compute on more tokens (the zero line; right is better): the mean over three training seeds of the per-seed difference, with whiskers of ± 1 sd.

Only the teacher's states as inputs and targets beat the compute-matched Transformer. At 135M, LIFT improves perplexity over the compute-matched Transformer by 0.47 ± 0.08 (Figure 5). Trained on its own states, the same architecture ties it ($+0.01 \pm 0.05$, within 0.07 at every seed); the teacher thus accounts for 98% of LIFT's gain over the compute-matched Transformer. Self-

generated states do improve on a Transformer trained on the same tokens, by 1.40 perplexity, but spending the extra pass on data instead improves on it by about as much, 1.39. With the teacher's states as inputs only ($\lambda = 0$), LIFT falls 0.19 ± 0.02 behind the compute-matched Transformer (31.23 ± 0.03 against 31.04 ± 0.00), and behind the no-teacher variant too. Without the alignment term the model makes less use of the teacher's states, reaching 30.72 ± 0.04 when fed them against 30.25 ± 0.05 for LIFT, and its own states drift further from them: feeding it its own states instead costs 0.51 perplexity, against 0.32 for LIFT, and run without states it is a full point behind the token-matched Transformer (33.47 against 32.44). The alignment term is thus what carries the teacher-forced channel over to inference (§3.2).

As a target only, the teacher is worth less than the data. Distillation falls behind the compute-matched Transformer by 0.77 ± 0.04. Its forward pass through the teacher is better spent on more tokens; used as both inputs and targets, the same teacher's states give LIFT its lead.

Tuned distillation. The Distillation baseline of Table 1 reuses LIFT's support and temperature: the top $1{,}024$ tokens with an aggregated tail, $\tau = 1.5$, $\lambda = 1.5$, next-token loss kept. Three further variants at 135M, with the same teacher, seed, data order and schedule, do not close the gap. The same baseline at $\tau = 1$ ties it (31.80 against 31.77). The recipe of Busbridge et al. (2025), a pure forward KL over the full vocabulary at $\tau = 1$ without the next-token loss, is behind even the token-matched Transformer (32.66 against 32.43, $[0.19, 0.29]$), and annealing that student with the next-token loss recovers only part of the gap (32.29). LIFT leads the best of the four by 1.11 ($[1.06, 1.17]$).

Downstream columns. At 135M and matched compute, most downstream differences are within seed noise, in the sense of Table 10 (overlapping mean $\pm$ sd ranges). Two are beyond it: LIFT's generative score (15.0 ± 0.6 against 13.6 ± 0.6) and Distillation's reference-completion loss (1.096 ± 0.006 against 1.085 ± 0.005 bits per byte). Without a teacher no column moves beyond noise; on arithmetic the no-teacher variant has LIFT's mean (2.28 bits per byte for both), and both are within the compute-matched Transformer's spread (2.39 ± 0.09).

Backpropagating through own states. Training on the model's own states exactly means backpropagating through the recurrence, as LIFT-RNN does on S_5 (§4). This makes training sequential over positions, removing the parallelism that pretraining relies on: each position waits for the state of the one before it, the backward pass traverses every step, and the attention inputs of every step must be stored, which grows quadratically with the sequence length in a direct implementation; activation checkpointing and fused attention kernels, built for parallel training, do not apply as they are. Feedback models therefore truncate the recurrence to a few parallel passes and backpropagate through them, as T^{2}MLR and the Multi-pass Transformer do. At matched compute both fall behind the vanilla Transformer, and both need more memory than LIFT (§5.3, Table 18). The no-teacher variant above is the cheapest point of this family, one extra pass without gradient, and ties the compute-matched Transformer. None of the variants that generate their own training states beats a Transformer given the same compute; LIFT, whose training states come from the teacher, does.

D.2 Channel Width

We vary k, the number of teacher tokens a state keeps (Eq. 4), at 135M, in both the fed-back state and the support of the alignment term, with everything else as in Table 1, including its seed. At $k = 1$ the state is the teacher's top token scaled by its probability, a second token channel; $k = 1024$ is our setting. Figure 6 shows each model with its states and, for reference, the same model run without them (LIFT w/o states in Table 1).

The gain grows with the width of the channel. Perplexity improves monotonically with k. The one-token channel is 1.16 behind the compute-matched Transformer and $k = 16$ is 0.39 behind; only $k \geq 128$ is ahead of it, by 0.14, by 0.38 at $k = 1024$ and by 0.52 at $k = 2048$. Measured against the token-matched Transformer, $k = 1$ keeps 13% of LIFT's gain, $k = 16$ keeps 56% and $k = 128$ keeps 86%. An extra token of input is thus not what LIFT's states contribute; their width is. Arithmetic follows the same order: 2.50, 2.35, 2.31 and 2.29 bits per byte at $k = 1$, 16,

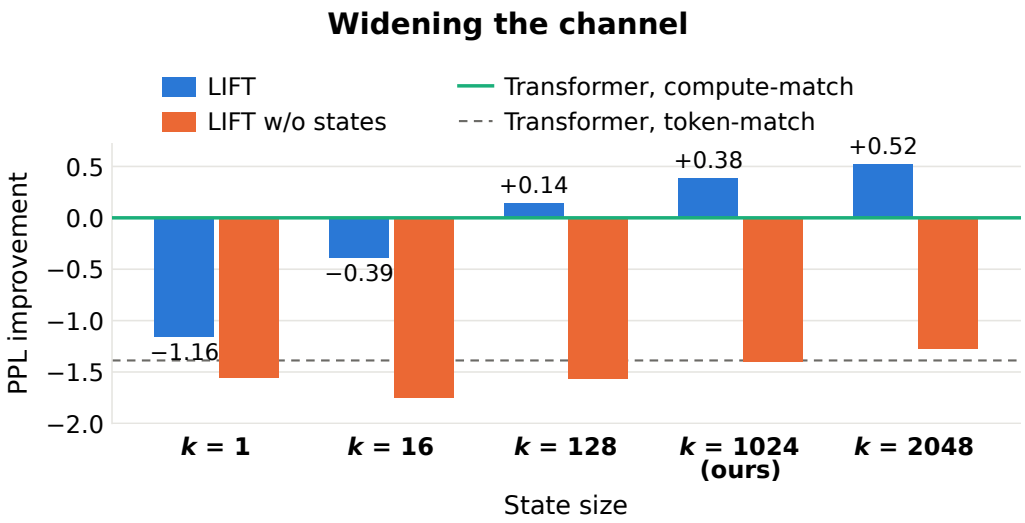


Figure 6: Channel width at 135M: improvement in perplexity over the compute-matched Transformer (the zero line; up is better) for k from 1 to 2048, with the model's own states fed back (LIFT) and for the same model run without states (LIFT w/o states). Dashed: the token-matched Transformer. Every step in k is beyond the paired bootstrap 95% interval over evaluation sequences (Table 6's protocol).

128 and 1024, the one-token channel at the level of the Transformers (2.46 token-matched, 2.50 compute-matched).

The width acts through the states. Run without states, the models trained at every k are within 0.48 perplexity of each other and within 0.36 of the token-matched Transformer: widening the channel, which also widens the alignment term's support, does not train a better Transformer. What grows with k is what the states carry at inference, 0.40 perplexity at $k = 1$, 1.36 at $k = 16$, 1.70 at $k = 128$, 1.78 at $k = 1024$ and 1.79 at $k = 2048$.

Choice of k. Returns diminish with width, from 0.77 perplexity between $k = 1$ and 16 and 0.53 between 16 and 128 to 0.24 between 128 and 1024, still well beyond evaluation noise (paired bootstrap 95% interval $[0.20, 0.29]$). Doubling to $k = 2048$ improves perplexity by a further 0.14 ($[0.10, 0.18]$; one training seed, against a seed spread of 0.08 for LIFT at 135M), but not through the states: what they carry has stopped growing (1.78 at $k = 1024$, 1.79 at $k = 2048$), and the model run without states improves by about as much (32.31 against 32.44), within the spread of the without-states models across k. The width costs almost nothing: k enters the compute only through the soft-token sum $E^\top \mathbf{s}_i$, 0.13% of a training step at 1B for $k = 1024$ (Table 17). We use $k = 1024$ at every scale.

D.3 Trunk vs. Adaptation Phase

The gain is built by the teacher-forced trunk, not by the adaptation phase. Two further runs isolate the adaptation phase of §3.2. *Adaptation only*: we take the token-matched Transformer at 90% of training, add a freshly initialized fusion layer, and run LIFT's adaptation phase over the last 10%. At 135M this reaches 32.53 perplexity, slightly worse than the Transformer itself (32.43); at 1B, 19.68 against 19.79, 12% of LIFT's gain over the Transformer (18.81). *No adaptation*: LIFT at 1B trained with the teacher until the end keeps 73% of that gain (19.07) and is still ahead of the compute-matched Transformer (19.17); adapting over the last 5% instead of 10% recovers the rest (18.81). The adaptation phase thus fits to the model's own states a channel that the teacher-forced trunk has already made useful. Run over the whole of training, it is the no-teacher variant above, which only ties the compute-matched Transformer.

E PREFILL MODES: REFINING THE PROMPT'S KEY-VALUE CACHE

During prefill nothing is generated: the pass over the prompt only builds the key-value cache that decoding attends to, and the state entering the first answer token. The evaluations of §5 build this cache sequentially, feeding each prompt position the state predicted at the preceding one. Here we ask whether the cache can be built in parallel instead, and at what price. Decoding is sequential in every mode, as it is in any autoregressive Transformer: each answer token is fed the state predicted at the preceding position (Eq. 4). Only the construction of the prompt's cache differs.

Prefill modes. **W/o states**: one parallel pass over the prompt with the learned bias $\mathbf{b}$ at every position after the first, as in a standard Transformer. ***R* refinements**: R further parallel passes over the prompt, each fed, at every position after the first, the state that the previous pass predicted at the preceding position; the first decoding step attends to the cache of the last pass. **Sequential**: the prompt is processed one position at a time, as the answer is; this is LIFT as evaluated in §5. The reference point is LIFT w/o states, which uses no states in prefill or decoding.

Relative gain. For each group g of MC, Gen. and RC, the *states gain* is $G_g = x_g^{\text{LIFT}} - x_g^{\text{w/o}}$, the difference between LIFT, with states through prefill and decoding, and LIFT w/o states. The relative gain of a mode or model X is $(x_g^X - x_g^{\text{w/o}})/G_g$, and we report its mean over the three groups: 0% is LIFT w/o states and 100% is LIFT.

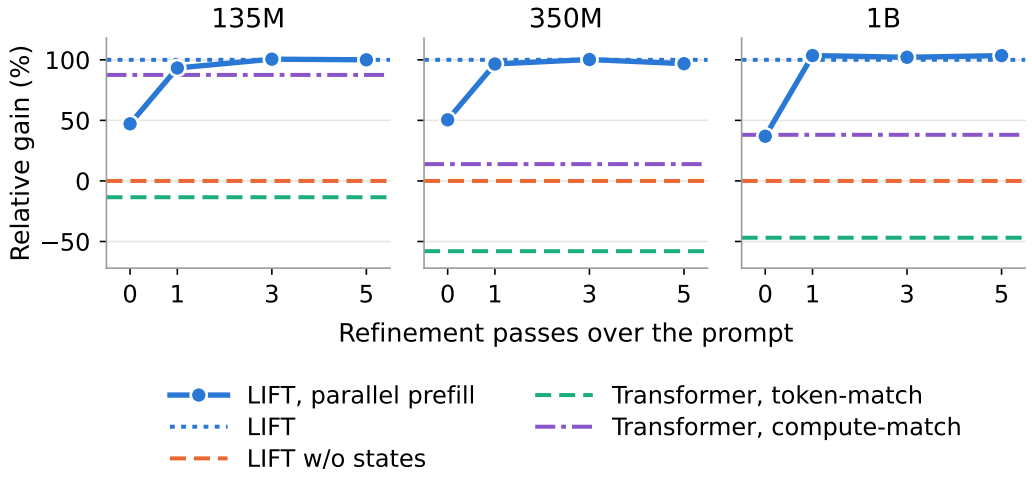


Figure 7: Relative gain of LIFT with parallel prefill (blue points) by the number of refinement passes over the prompt, against LIFT (100%) and LIFT w/o states (0%). The token- and compute-matched Transformers are placed on the same scale; below 0%, a model is behind LIFT w/o states.

One refinement pass suffices for the key-value cache. With one refinement pass, LIFT matches sequential prefill to within 0.2 points on MC, Gen. and RC at every scale, a relative gain of 93–104%; three and five passes change the scores by at most 0.2 points (Figure 7, Table 16). Arithmetic, reported separately in Table 16, follows the same pattern: one refinement recovers 89–102% of its states gain, and three passes essentially all of it. The Full-bandwidth Transformer shows the same pattern for its fused prefill passes, where most of the improvement comes from the first one (Wang et al., 2026). Refining the prompt's cache thus costs one extra parallel forward pass rather than n sequential steps.

Without refinement, LIFT stays ahead of the Transformer baselines. Prefilling without states and decoding with them keeps a relative gain of 37–51%, at the prefill cost of a standard Transformer. In this mode LIFT is ahead of the token-matched Transformer at every scale, as LIFT w/o states already is, and it is ahead of the compute-matched Transformer at 350M and level with it at 1B (Figure 7). On arithmetic this mode keeps less of the gain, 18–26%; it is still ahead of the compute-

matched Transformer at every scale and of the token-matched one at 135M and 350M, and level with the latter at 1B.

Cost. Prefill without states has no sequential dependency along the prompt: it is one parallel pass. With R refinements, positions depend on each other only across consecutive passes, so the prompt is read in $R+1$ parallel passes at $R+1$ times the FLOPs of one. Sequential prefill costs the FLOPs of one pass, since the key-value cache avoids recomputation, but reads a prompt of n tokens in n dependent steps. A refined cache can thus be obtained either with one extra parallel pass, doubling the prefill FLOPs, or with n sequential steps at the FLOPs of one pass. Counting every forward pass, over the prompt or over one answer token, as one step, an answer of D tokens takes $D+1$ steps without refinement and $D+2$ with one: the extra pass adds $1/(D+1)$ of the steps, a share that shrinks as the answer grows. At the mean answer lengths of our generative tasks, it adds 17% on TriviaQA (5 tokens), 13% on the generative QA suite (6.5 tokens) and 0.5% on chain-of-thought BBH (209 tokens).

Table 16: Prefill modes of LIFT on arithmetic and the OLMES groups of Table 1, with the token- and compute-matched Transformers of that table. Prefill cost: forward passes to read a prompt of n tokens, and FLOPs relative to one forward pass over the prompt. Full propagation: every answer token is fed the state predicted at the preceding position; –: a model without states. The prefill, decoding and cost columns describe how each model runs in deployment, where the answer is generated one token at a time. They are not the cost of scoring these benchmarks, where a given answer, as on MC and RC, is scored in one parallel pass by the models without states.

				Prefill cost					
	Model	Prefill	Decoding	passes	FLOPs	Arith. BPB ↓	MC acc. %	Gen. %	RC BPB ↓
135M	LIFT	w/o states	w/o states	1	1×	2.42	38.7	12.0	1.105
	LIFT	w/o states	full propagation	1	1×	2.40	38.9	13.9	1.093
	LIFT	1 refinement	full propagation	2	2×	2.30	39.3	15.3	1.079
	LIFT	3 refinements	full propagation	4	4×	2.29	39.4	15.5	1.078
	LIFT	5 refinements	full propagation	6	6×	2.29	39.4	15.5	1.078
	LIFT	sequential	full propagation	n	1×	2.29	39.4	15.5	1.078
	Transformer, token-matched	–	–	1	1×	2.46	38.1	13.6	1.110
	Transformer, compute-matched	–	–	1	1×	2.50	39.5	14.1	1.080
350M	LIFT	w/o states	w/o states	1	1×	2.21	44.2	21.0	0.945
	LIFT	w/o states	full propagation	1	1×	2.19	44.9	21.7	0.936
	LIFT	1 refinement	full propagation	2	2×	2.12	45.0	23.2	0.922
	LIFT	3 refinements	full propagation	4	4×	2.12	45.1	23.1	0.921
	LIFT	5 refinements	full propagation	6	6×	2.12	45.0	23.0	0.922
	LIFT	sequential	full propagation	n	1×	2.12	45.0	23.2	0.921
	Transformer, token-matched	–	–	1	1×	2.46	43.2	21.0	0.956
	Transformer, compute-matched	–	–	1	1×	2.33	44.1	22.0	0.942
1B	LIFT	w/o states	w/o states	1	1×	2.05	51.2	32.9	0.791
	LIFT	w/o states	full propagation	1	1×	2.03	51.6	33.9	0.784
	LIFT	1 refinement	full propagation	2	2×	2.00	52.5	35.2	0.772
	LIFT	3 refinements	full propagation	4	4×	1.99	52.5	35.1	0.772
	LIFT	5 refinements	full propagation	6	6×	1.99	52.5	35.2	0.772
	LIFT	sequential	full propagation	n	1×	1.99	52.5	35.0	0.772
	Transformer, token-matched	–	–	1	1×	2.03	50.5	32.9	0.808
	Transformer, compute-matched	–	–	1	1×	2.07	51.6	34.8	0.792

F Cost Analysis

F.1 Training and Inference FLOPs

Table 17 itemizes every addition LIFT makes to a vanilla Transformer's compute, excluding the teacher's forward pass, for the 1B model ($d = 2048$, $L = 16$, $N = 1.07$B non-embedding parameters, $k = 1024$, $|\mathcal{V}| = 100{,}352$). A multiply-add counts as two FLOPs. Percentages are relative to the standard $6N$ FLOPs per training token ($2N$ forward, $4N$ backward) over non-embedding parameters; counting the output head and the attention scores at 4k context lowers every percentage by about a quarter. The scaling column uses $N \approx 16Ld^2$, the OLMo 2 block shape at 1B ($N \approx 20Ld^2$ at 32B, where the fusion block is 0.85%).

Three remarks. First, the KL term needs the student's full-vocabulary log-partition at temperature τ, one extra pass over logits that the language-modeling loss already materializes; no $(B, T, |\mathcal{V}|)$

Table 17: Compute added per token by LIFT at 1B, excluding the teacher forward. The trunk is the first 90% of training; the adaptation phase is the last 10% (§3.2).

Component	FLOPs / token	1B (MFLOPs)	% of a $6N$ step	Scaling
Fusion block, Eq. (5), forward + backward	$66d^2$	277	4.3%	$11d^2/N = 11/(16L)$, falls with depth
Soft-token sum $E^\top \mathbf{s}_i$, forward + backward	$4kd$	8	0.13%	$k/(24Ld)$
Forward KL on the top-k support + tail, forward + backward	$\approx 5\lvert\mathcal{V}\rvert + 20k$	0.5	$< 0.01\%$	$\approx 5\lvert\mathcal{V}\rvert/(96Ld^2)$
Prefix state dropout, learned bias	0	0	0	–
Trunk total, per training token		286	4.4%	
Adaptation phase: 1.5 extra no-gradient forwards on average	$1.5\,(2N + 22d^2 + 2kd)$	3,366	$+52\%$ of those steps	fixed by the schedule
Whole run ($0.9\times$ trunk $+ 0.1\times$ adaptation)			$\approx 9.6\%$	
Inference, per generated token	$22d^2 + 2kd$ + top-k	97	4.5% of a $2N$ forward	$11/(16L)$

tensor is created beyond them. Second, the adaptation passes run without gradient and their feedback distribution is detached, so they hold no activation memory; a step in that phase costs about $1.5\times$ a trunk step. Third, over a whole run the additions total under 10% of the vanilla training FLOPs, and every term except the schedule-fixed adaptation shrinks with model size.

Compute-matched budgets. The compute-matched Transformers receive the training FLOPs of the whole LIFT run, including the teacher's forward pass, the fusion layer and the extra passes of the adaptation phase: 19.3B, 49.9B and 152.4B tokens at 135M, 350M and 1B, i.e., 1.42–$1.44\times$ LIFT's budget. This accounting also charges a teacher forward pass during the adaptation phase, which our implementation mistakenly ran although its output is not used there (§A.3); without it, the matched budgets would be 18.8B, 48.7B and 148.8B tokens. The compute-matched baselines thus train on about 2.5% more tokens than the method requires, which favors them.

F.2 Memory Footprint

Table 18 reports the peak GPU memory of LIFT, T^{2}MLR and the Multi-pass Transformer in the setting of §5.3, measured with the same code, in the same container and on the same hardware. Setup: 2 NVIDIA B200 GPUs (183 GB each) with data parallelism, PyTorch 2.9 (NVIDIA container 25.10), bf16, sequences of 2,048 tokens, a global batch of 256 sequences reached by gradient accumulation, and the production configuration of each method: T^{2}MLR(13,18) with 16 forward and 4 backward refinement passes, and LIFT in its trunk configuration, including the teacher forward pass and the top-k ($k = 1024$) state construction, with the forward-KL term enabled. The latter makes this a conservative memory comparison for the KL-free model of §5.3. Peak memory is the maximum memory allocated by PyTorch on one GPU.

LIFT's memory beyond a vanilla step includes the teacher's logits for the top-k selection and the dense soft-token matrix, both transient. T^{2}MLR stores the recurrent cache and the activations of the refinement passes it differentiates through. The Multi-pass Transformer has the largest peak memory because its three-pass batches retain activations from all three passes.

Table 18: Peak memory per GPU of LIFT, T^{2}MLR and the Multi-pass Transformer at 135M (SmolLM2 backbone, 2,048-token sequences, B200). Memory is the peak allocated by PyTorch. The Multi-pass Transformer's peak is measured on a three-pass batch, which determines the memory needed to run its full schedule.

Model	Micro-batch / GPU	Peak memory
LIFT	8	**35.4 GB**
T^{2}MLR(13,18)	8	44.7 GB
Multi-pass Transformer	8	62.9 GB
LIFT	4	**18.5 GB**
T^{2}MLR(13,18)	4	23.0 GB
Multi-pass Transformer	4	32.1 GB

G Extended Related Work

Depth and sequential computation. Looped and recurrent-depth Transformers reapply a block of layers to increase the computation spent on each token (Dehghani et al., 2019; Saunshi et al., 2025; Geiping et al., 2025); their inference cost grows with the number of iterations, and the depth available to a token is still bounded by the iterations run at that step. Chain of thought instead adds serial computation through generated tokens (Wei et al., 2022), which extends the expressivity of fixed-depth Transformers (Feng et al., 2023; Li et al., 2024; Merrill & Sabharwal, 2024), and test-time methods scale it further through sampling, verification and search (Wang et al., 2023; Lightman et al., 2024; Snell et al., 2025). Every such step, however, passes through the single decoded token. In LIFT, the computation's depth grows with the sequence at a fixed per-token cost (§F.1), and the approach is complementary to both.

Teachers as targets and as inputs. Pretrained teachers usually serve as targets: knowledge distillation matches their distributions (Hinton et al., 2015; Kim & Rush, 2016), also in LM pretraining (Gemma Team, 2024), and richer objectives supervise future tokens or latents (Gloeckle et al., 2024; Zhang et al., 2026), leaving inference unchanged. An LM's distribution has also been fed to another model as an input, but one that remains at inference (Sriram et al., 2018). In LIFT, the teacher's distribution is a training input that the model learns to replace with its own, and our distillation baseline isolates the target role (§5.1). As with born-again and weak-to-strong students (Furlanello et al., 2018; Burns et al., 2024), the student can surpass its teacher (§4); our adaptation phase is scheduled sampling (Bengio et al., 2015; Mihaylova & Martins, 2019) on the state channel.

Comparison with the closest methods. Table 19 lists, for the methods closest to LIFT, what is fed back, where it enters the model, whether the sampled token is kept, how the fed-back states are obtained during training, and at which stage the channel is learned. LIFT is the only method whose training-time states are supplied by an external pretrained LM, which is what allows a single parallel pass during pretraining.

Table 19: Methods that feed information back across generation steps. *Training-time states*: how the fed-back states are obtained during training; *own* means generated by the model being trained. †arXiv preprint.

Method	Fed-back signal	Enters at	Token	Training-time states	Stage
Feedback Transformer (Fan et al., 2021)	All layers' states of past positions	Attention of every layer	kept	own, sequential	pretraining
RMT (Bulatov et al., 2022)	Output memory tokens of the previous segment	Input of the next segment	kept	own, sequential over segments	pretraining
LCKV (Wu & Tu, 2024)	Top-layer keys and values	Attention of every layer	kept	own, iterative parallel passes	pretraining
Turbo Connection (Tang & Lu, 2026)	Higher-layer states of the previous token	Lower layers	kept	own, sequential in groups	fine-tuning
PonderLM (Zeng et al., 2026b)	Top-k soft token of its own prediction	Input of the same position	kept	own, extra passes per position	pretraining
PonderLM-2 (Zeng et al., 2026a)	Last hidden state	An added input position	kept	own, Jacobi passes	pretraining
T^2MLR† (Cai et al., 2026)	Middle-layer hidden state	An earlier middle layer	kept	own, Jacobi passes	pretraining
Full-bandwidth Transformer† (Wang et al., 2026)	Top-layer hidden state	Input, gated with the token	kept	own, one to three parallel passes	pretraining
Latent Recurrent Transformer† (Huang et al., 2026)	Hidden state of a source layer	Attention and residual stream	kept	own, interleaved parallel training	pretraining
WhiteMatter† (Zhang & Ren, 2026)	Keys and values of all layers	Attention of every layer	kept	own, fixed-point passes	pretraining
Coconut (Hao et al., 2025)	Last hidden state	Input	replaced	own, sequential	post-training
CODI (Shen et al., 2025)	Projected last hidden state	Input	replaced	own, sequential	post-training
PCCoT (Wu et al., 2025)	Last hidden state	Input	replaced	own, Jacobi passes	post-training
Soft Thinking, MoI (Zhang et al., 2025; Zhuang et al., 2025)	Soft token of its own distribution	Input	replaced, mixed	none (training-free)	inference only
HRPO (Yue et al., 2025)	Soft token of its own distribution	Input, gated with the token	kept	own, RL rollouts	post-training
CoT2 (Gozeten et al., 2026)	Soft token	Input	replaced	teacher-forced, from an oracle	task training
CoLaR (Tan et al., 2025)	Compressed reasoning-token embeddings	Input	replaced	teacher-forced, from gold traces	post-training
Thinking States† (Amos et al., 2026)	Compressed reasoning of the previous chunk	A shallow layer	kept	teacher-forced, from gold annotations	post-training
CoCoMix (Tack et al., 2026)	Its own predicted concept vector	Interleaved with hidden states	kept	own, same pass	pretraining
SMT† (Kumar & Isola, 2026)	RNN memory	RNN state	–	teacher-forced, from a jointly trained encoder	pretraining (RNN)
LIFT (ours)	Top-k next-token distribution	Input, fused with the token	kept	teacher-forced, from a pretrained LM	pretraining