TURNOUT: ADAPTIVE ATTENTION SWITCHING FOR EFFICIENT REASONING

Anonymous authors
Paper under double-blind review

ABSTRACT

Post-hoc transformer linearization replaces the key-value cache of full attention, which grows linearly with generation length, with a fixed size state. This change reduces memory consumption but often hurts accuracy. In reasoning tasks, which require long generations, this effect is especially pronounced. In this work we introduce TURNOUT, a method that mitigates accuracy loss as a result of linearization, while maintaining its memory-reduction benefits. We apply post-hoc linearization to a pretrained full-attention model, training a linear path through distillation while leaving the original full-attention path unchanged. During response generation, the resulting hybrid model retains both paths and uses a learned policy to control when each layer switches from full to linear attention. We train the policy with reinforcement learning using a reward that balances answer correctness and memory use. Empirical evaluation of our method shows that the learned adaptive policy achieves a better accuracy-efficiency Pareto frontier on reasoning tasks compared to fixed switching and static hybrid baselines.

1 INTRODUCTION

Transformer language models remain the dominant architecture for high-quality sequence modeling, but their inference cost grows linearly with context length. During decoding, every generated token attends to an expanding key–value (KV) cache, making long-context generation increasingly memory intensive and limiting throughput once cache movement dominates the runtime (Xiao et al., 2023; Dao, 2023; Kuzina et al., 2026). Linear attention mechanisms offer an attractive alternative: they compress past context into a fixed-size state, so the memory footprint no longer grows linearly with the number of generated tokens (Choromanski et al., 2020; Yang et al., 2023; Sun et al., 2023; Yang et al., 2024; 2025b; Kimi Team et al., 2025).

Post-hoc linearization is appealing because it starts from an already trained softmax-attention backbone and converts its attention layers into cheaper recurrent alternatives (Mercat et al., 2024; Zhang et al., 2024a; Lan et al., 2025; Kuzina et al., 2026), instead of training a linear model from scratch. However, directly replacing softmax attention introduces a substantial mismatch with the pretrained model and often reduces accuracy on tasks that rely on precise retrieval or multi-step reasoning (Mercat et al., 2024; Zhang et al., 2024a; McDermott et al., 2025; Kuzina et al., 2026).

This paper focuses on hybrid post-hoc linearization, in which the original multi-head attention (MHA) path remains available alongside a linearized alternative. The two paths share the pretrained backbone and differ only in their attention-specific components. As such, each transformer layer in our model can switch from MHA to linear during generation by generating a linear state from previously generated key and value vectors. We use MHA for the prompt, after which a learned policy decides when each layer switches to linear attention during generation. This matters because the best attention mechanism can depend on the phase of inference. For example, MHA could be preferable during prefill, where the model consumes the prompt and builds a representation of the context, while a linear recurrent path is much cheaper during long decoding, where the key-value (KV) cache would otherwise continue to grow. Jet-Nemotron (Gu et al., 2025) further shows that the importance of full-attention layers is task-dependent: layers that are critical for MMLU are not necessarily important for math problems and vice versa.

For the linear path, we build on the design of Kuzina et al. (2026), which reports strong post-hoc linearization results on common reasoning benchmarks and MMLU. We improve their recipe with

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

distillation, as supported by prior work on transforming attention-based models to recurrent or hybrid alternatives (Ro et al., 2025; Wang et al., 2024; Bick et al., 2025). Furthermore, we combine LoRA adaptation of the attention output projection with short convolutional mixing. As a result, we obtain a linearized model that outperforms all previously reported post-hoc linearized models, as well as pure sliding-window replacements (Jolicoeur-Martineau et al., 2026).

We then design a system that can decide, for each layer at each generated token, whether to keep a layer as MHA or switch to linear, simultaneously trying to maximize accuracy while minimizing memory consumption. Note that, unlike prior work, our method not only decides *which* layers (layer dimension) to switch to linear, but also *when* to do so (token dimension). This flexibility is motivated by the observation that the relative benefits of full and linear attention can vary across tasks and may also depend on the difficulty of individual prompts, making a fixed switching schedule potentially suboptimal. We formulate this as a reinforcement learning (RL) problem. The policy receives both an accuracy-oriented reward and a memory-aware reward, so it is encouraged to preserve task performance while reducing inference memory. Unlike a fixed manual rule, the learned policy adapts the switch point to each input.Across the five reasoning benchmarks, the learned router scores 82.2% average accuracy, improving over the best manual fixed-switch baseline by 7.7 percentage points while reducing cache memory by 25.4% relative to full MHA.

Our main contributions are as follows:

- We improve post-hoc linearization accuracy on reasoning benchmarks.
- We propose TURNOUT, an architecture which enables inference-time switching between MHA and linear attention without training a separate model for each hybrid schedule.
- We formulate attention switching as an RL problem with memory-aware and accuracy rewards, and compare learned switching against manual fixed-switch baselines.

2 RELATED WORK

Linear-time sequence models. Linear-time sequence models provide the primitives used for post-hoc transformer linearization. Kernelized attention methods such as Performer (Choromanski et al., 2020) and feature-map approaches such as Hedgehog/Porcupine (Zhang et al., 2024b) approximate the softmax attention computation directly. A complementary line develops recurrent or state-based replacements, including Gated Linear Attention (GLA) (Yang et al., 2023), Delta Rule (Yang et al., 2024), RetNet (Sun et al., 2023), Longhorn (Liu et al., 2024), Gated Delta Networks (GDN) (Yang et al., 2025b), Kimi Linear (Kimi Team et al., 2025), and transformer–SSM views of Mamba-style models (Dao & Gu, 2024; Egorov et al., 2025). These models show that recurrent state updates can support efficient long-context computation, but they are usually proposed as standalone architectures or pretraining recipes. Our work instead starts from an existing pretrained softmax model and treats linear attention as one path inside a hybrid inference system.

Post-hoc linearization of pretrained transformers. Post-hoc linearization methods try to close the quality gap introduced by replacing softmax attention in an already pretrained transformer with a cheaper recurrent update. LoLCATs (Zhang et al., 2024a) and Liger (Lan et al., 2025) combine linearized attention with local exact attention such as sliding-window attention, while LoLA (McDermott et al., 2025), STILL (Meng et al., 2026), and Lizard (Nguyen et al., 2025) preserve selected exact tokens through sparse caches or hybrid token selection. Other approaches improve the conversion through adaptive distillation (Ro et al., 2025), direct layerwise linearization (Mercat et al., 2024), and low-rank adaptation of attention projections (Zhang et al., 2024a; McDermott et al., 2025). Most closely related to our work, Kuzina et al. (2026) analyze post-hoc transformer linearization in a frozen-backbone setting and report the strongest results among prior post-hoc linearization methods on common reasoning benchmarks and MMLU. We use their linear attention design as the base linear path in our hybrid model. In addition, we show that for harder reasoning benchmarks, cross-entropy training alone is not sufficient to recover the original model behavior, and we therefore use distillation to improve the linearized path before learning when to switch between MHA and linear attention.

Hybrid models and switching. Sequence-level hybrid models have shown that attention and recurrent computation can be used at different positions within a sequence. TransMamba (Li et al.,

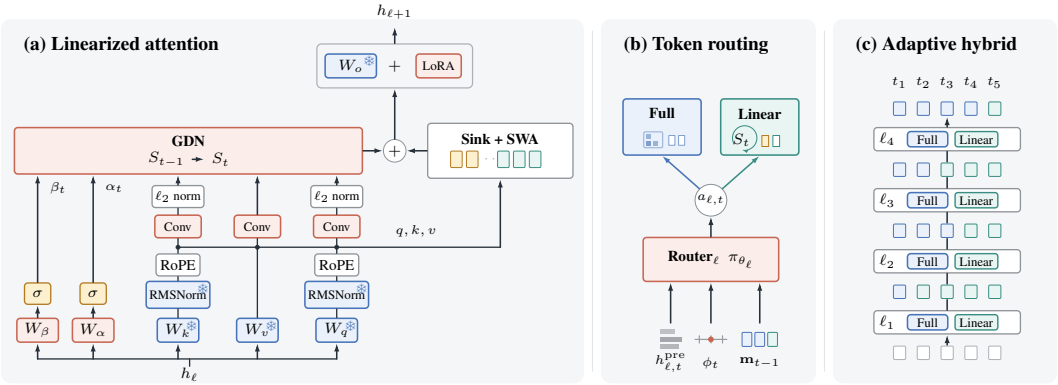


Figure 1: Overview of TURNOUT: linearized attention, token-level routing, and adaptive layerwise switching. **(a)** Frozen pretrained modules (blue snowflakes) are retained. Trainable gates, short convolutions, GDN, and LoRA form the linearized branch; exact Sink+SWA runs in parallel. **(b)** The router at layer ℓ uses the pre-attention hidden state, position and break-even features, and the preceding token's routing mask to select exactly one attention path. **(c)** Across a token sequence, different layers can switch from full attention (blue) to linear attention (teal) at different times.

2026b) assigns each layer a predefined position at which it switches from attention to Mamba. Oryx (Li et al., 2026a) shares most parameters between attention and recurrent mixers and supports switching during prefill and generation, but it pretrains the hybrid model with mixed-mode sequences and selects the mixer schedule manually. These methods establish the feasibility of temporal switching, but their fixed schedules cannot adapt to the input. We instead start from a post-hoc linearized pretrained model and learn an input-dependent switch point for each layer, using fixed switching only as a baseline.

3 LINEARIZATION

To isolate linearization quality, we first evaluate the linear path without a switching policy. We use Qwen3-8B (thinking model) (Yang et al., 2025a) and Llama 3.1-8B-Instruct (Grattafiori et al., 2024) as teachers. Our main evaluation focuses on five reasoning benchmarks: GSM8K (0-shot) (Cobbe et al., 2021), MATH-500 (Hendrycks et al., 2021b), GPQA-Diamond (Rein et al., 2023), HumanEval+, and MBPP+ (Liu et al., 2023). In this section, we also report two non-reasoning benchmarks, MMLU (5-shot) (Hendrycks et al., 2021a) and LAMBADA (OpenAI) (Paperno et al., 2016), to compare with prior linearization studies that do not report reasoning benchmarks. We omit other common reasoning benchmarks because they are too easy and can be solved by sliding-window attention (SWA) alone. Our linearization is based on Key to Going Linear (KtGL) (Kuzina et al., 2026). We improve the performance of their method by using a different training objective, dataset, and longer training.

Linear path and Training Objective The linear path combines a Gated Delta Network (Yang et al., 2025b) with a fixed 64-token SWA and 8 sink tokens. We train short convolutions and a rank-128 LoRA adapter on output projection (see Fig. 1a). We test three objectives used in prior linearization work: next-token cross-entropy, as used by Liger and KtGL (Lan et al., 2025; Kuzina et al., 2026); logit distillation, which matches the student's next-token distribution to the teacher's, as used by Mamba in the Llama and Llamba (Wang et al., 2024; Bick et al., 2025); and activation distillation, which matches intermediate attention outputs, as used in the attention-transfer stage of LoLCATs (Zhang et al., 2024a). The objective ablation in Table 3 (App. A.1) shows that logit distillation gives most of the improvement on reasoning benchmarks. Combining logit and activation distillation gives the best GSM8K result. The adapter ablation shows that LoRA on o_{proj} is important for reasoning: rank 128 substantially improves over the no-LoRA variant, while increasing the rank to 256 provides no further gain. We therefore use the mixed distillation objective with rank-128 LoRA. Formal definitions of the losses and the full ablations are provided in App. A.1.

Training Dataset In App. A.2, we compare training on DCLM-Edu (Li et al., 2024), OpenThoughts3 (OT) (Guha et al., 2025), and DCLM-Edu followed by OT. OT training for 4.5B

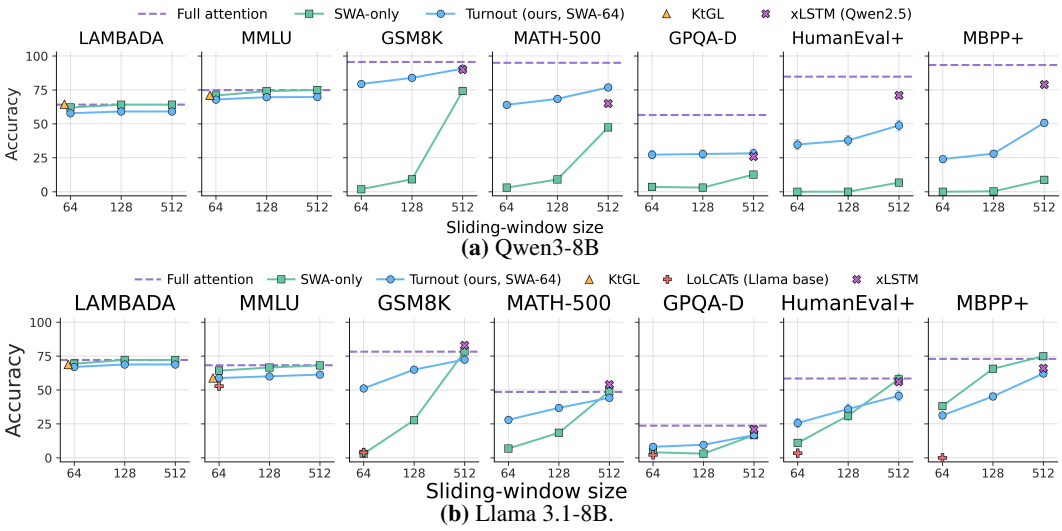


Figure 2: Accuracy across sliding-window sizes for (a) Qwen3-8B and (b) Llama 3.1-8B-Instruct. TURNOUT is trained with SWA-64 and evaluated with the three plotted windows. Dashed lines show base model accuracy. KtGL (Kuzina et al., 2026), LoLCATs (Zhang et al., 2024a), and xLSTM (Hauzenberger et al., 2026) use the values reported in the original papers.

tokens produces the strongest thinking results among the trained linearized models. We therefore use this setup as the linearized Qwen3 model and apply the same training recipe to Llama.

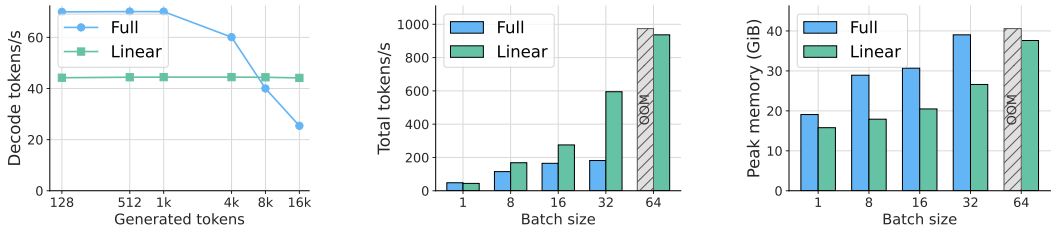
Sliding-Window Size As mentioned above, our linear path combines linear attention with sliding-window attention and sink tokens. Based on non-reasoning evaluations, recent work has shown that SWA with sink tokens alone can produce strong results, challenging the need for post-hoc linearization (Jolicoeur-Martineau et al., 2026). We verify whether this holds for reasoning benchmarks by comparing a model using only SWA and sink tokens with our linearized model. Notably, we use a sliding-window size of 64 during training and increase the SWA size to 128 and 512 only during evaluation. We keep the number of sink tokens fixed at 8 throughout.

Figure 2 reports accuracy across seven benchmarks. For both Qwen3-8B and Llama 3.1-8B, increasing the SWA window recovers full-attention performance on LAMBADA and MMLU. Reasoning benchmarks show a different pattern. At SWA-64, training the linear path improves almost every result across the two models. For Qwen3-8B, the linearized model remains stronger on all five reasoning benchmarks even at SWA-512. For Llama 3.1-8B, a 512-token SWA window closes much of the gap and matches full attention on several reasoning benchmarks. The exact values for all window sizes are reported in Appendix A.3.

4 ADAPTIVE HYBRID ROUTING

Full attention and linear attention offer complementary advantages during generation. Full attention is faster for short sequences, whereas linear attention maintains a fixed-size state that is more efficient for long sequences. To illustrate this trade-off, we profile the full-attention and linearized Qwen3-8B models on the same subset of 128 MATH-500 examples (Fig. 3). At batch size 1, full attention achieves higher token throughput for short sequences, but linear attention becomes faster as the sequence grows (Fig. 3a). Linear attention also uses less peak memory (Fig. 3c), allowing larger batches and therefore higher aggregate throughput (Fig. 3b). These complementary regimes motivate TURNOUT, which uses full attention at the beginning of generation and switches to linear attention as the response grows longer.

The post-hoc linearization procedure described in the previous section, makes this transition possible. It preserves the original full-attention path and adds a linear path to each layer which shares most of the parameters with MHA. We first describe layerwise switching in Sec. 4.1, then introduce a lightweight router that decides when each layer should switch (Sec. 4.2) and propose to train it with



(a) Throughput by length **(b)** Throughput by batch size **(c)** Peak allocated memory

Figure 3: Inference profiling of the full-attention and linearized Qwen3-8B models on an H100 GPU using the same fixed subset of 128 MATH-500 examples. (a) Throughput across different sequence legths at batch size 1. (b) Throughput at batch sizes 1, 8, 16, 32, and 64. (c) Peak PyTorch allocated memory at the same batch sizes.

reinforcement learning (Sec. 4.4). Figure 1 provides an overview of our proposed method, TURNOUT, showing the two attention paths, the router, and the layer-specific switch points.

4.1 LAYERWISE SWITCHING

Let P be the prompt length, G the number of generated tokens, and $T = P + G$. For each layer $\ell \in \{1, \ldots, L\}$, let $s_\ell \in \{0, \ldots, G\}$ be the number of generated tokens processed with full attention and let $\tau_\ell = P + s_\ell$ be the corresponding total-token switch position. The routing mask is

$$m_{\ell,t} = \begin{cases} 1, & t \le \tau_\ell, \\ 0, & t > \tau_\ell, \end{cases} \tag{1}$$

where $m_{\ell,t} = 1$ means that layer ℓ uses full attention to process token t and $m_{\ell,t} = 0$ means that it uses linear attention. Thus $s_\ell = 0$ switches immediately after the prompt, while $s_\ell = G$ keeps the layer in full attention for the complete answer.

Full attention stores a KV-cache that grows with the sequence. Linear attention stores a fixed-size recurrent state and the local Sink+SWA state described in Section 3. After a layer switches, unused KV cache is released from the memory. As a deterministic baseline, fixed switching may use the same generated-token budget for every layer and example, i.e. $s_\ell = s_{\text{fixed}}$, for all ℓ, where the budget s_{fixed} is selected to satisfy the target KV-cache constraint.

4.2 PER-LAYER ROUTER

TURNOUT replaces the fixed budget with layer- and input-dependent decisions. Before the attention module of each layer that remains in full-attention mode, the router predicts whether that layer should remain in full attention or transition to linear attention. The router combines two signals: a *schedule score* based on the token position and a *context score* based on the hidden representation of the specific prompt. The stay-full logit and probability are

$$\nu_{\ell,t} = \nu_{\ell,t}^{\text{schedule}} + \nu_{\ell,t}^{\text{context}}, \qquad p_{\ell,t}^{\text{full}} = \sigma(\nu_{\ell,t}). \tag{2}$$

where σ denotes the sigmoid function.

Schedule Score. The schedule score uses position and routing information available before the layer is evaluated. We define the memory break-even point $B = M_{\text{linear}}/c_{\text{KV}}$, where M_{linear} is the cache memory used by one layer's linear path and c_{KV} is the memory used by one layer's full-attention KV-cache per token. Thus B is the number of full-attention tokens whose KV-cache uses the same memory as the linear-path cache. It can be computed directly from the memory footprints of the two paths or treated as a tunable reference point that biases the router toward switching after approximately B total tokens. The same break-even point defines the memory reward in Section 4.4; we therefore provide the router with position features relative to B, aligning its inputs with the efficiency objective used to train it. The schedule input is

$$\boldsymbol{\phi}_t = \left[\boldsymbol{\psi}(t); \frac{t}{B}; \text{clip}\left(\frac{t}{B}, 0, 1\right); \max\left(\frac{t-B}{B}, 0\right); \mathbf{m}_{t-1} \right], \qquad \nu_{\ell,t}^{\text{schedule}} = \text{MLP}_\ell(\boldsymbol{\phi}_t), \tag{3}$$

where $\psi(t)$ is a log-spaced sinusoidal encoding of the total token position and $\mathbf{m}_{t-1}$ records which layers were still using full attention at the preceding token. Each layer has an independent small MLP that maps these features to its schedule score.

Context Score. The context score enables input-dependent switching decisions. It uses the current pre-attention hidden state $\mathbf{h}_{\ell,t}$ as a frozen input, so router training does not update the language model. A shared projection produces a compact context representation, followed by a layer-specific linear layer with scalar output:

$$\mathbf{u}_{\ell,t} = \mathrm{SiLU}(W^c\,\mathrm{sg}[\mathbf{h}_{\ell,t}] + \mathbf{b}^c)\,, \qquad \nu^{\mathrm{context}}_{\ell,t} = \mathrm{Linear}_\ell(\mathbf{u}_{\ell,t}). \tag{4}$$

Let θ denote all trainable parameters of the schedule and context branches. The routing policy is the Bernoulli distribution $\pi_\theta = \mathrm{Bernoulli}(p^{\mathrm{full}}_{\ell,t})$. For each layer that is still using full attention, the router samples $m_{\ell,t} \sim \pi_\theta$, with

$$m_{\ell,t} = 1 \;\Rightarrow\; \text{full attention}, \qquad m_{\ell,t} = 0 \;\Rightarrow\; \text{switch to linear attention}. \tag{5}$$

After the first zero, the mask remains zero and is no longer sampled for that layer. Figure 1b summarizes its inputs and decision. We describe router architecture in more details in App. B.1.

4.3 Accuracy and Efficiency Reward

The router is trained to reduce cache memory while preserving answer quality. We use two ways to combine accuracy and efficiency:

$$R_{\mathrm{add}} = a(\hat{y}, y) + \frac{\alpha}{L}\sum_{\ell=1}^{L} e_\ell, \qquad R_{\mathrm{scaled}} = a(\hat{y}, y)\left(1 + \frac{\alpha}{L}\sum_{\ell=1}^{L} e_\ell\right). \tag{6}$$

Here $\hat{y}$ is the generated response and y is the reference answer. The accuracy score $a(\hat{y}, y)$ is one when $\hat{y}$ matches y and zero otherwise. The score $e_\ell \in [0, 1]$ measures the efficiency of layer ℓ, and α controls the trade-off between accuracy and efficiency. The two rewards differ only for an incorrect answer: R_{add} keeps the efficiency term, while R_{scaled} is zero. The latter option does not let the model optimize efficiency at the expense of accuracy. Figure 4 shows the two choices for e_ℓ defined below.

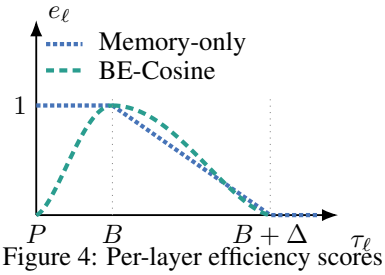


Figure 4: Per-layer efficiency scores when $P < B$. A layer with $\tau_\ell = P$ switches immediately after the prompt. The memory-only score is one for switch positions between P and B. The Break-even-cosine score increases from zero at P to one at B. Both scores are zero at $B + \Delta$.

We compute e_ℓ from the layer's peak cache memory C_ℓ, measured in equivalent KV-cache tokens. Because the switch position τ_ℓ counts both prompt and generated tokens, a layer that switches at τ_ℓ has a KV cache of τ_ℓ tokens. After the switch, its linear-path cache uses the same memory as B KV-cache tokens. Thus, $C_\ell = \max(\tau_\ell, B)$. If the layer never switches, only the KV-cache is used and $C_\ell = T$. The decay parameter $\Delta > 0$ sets the range over which the score falls from one at $C_\ell = B$ to zero at $C_\ell = B + \Delta$.

We use two definitions of e_ℓ. The *memory-only* score depends only on C_ℓ. Switching before break-even point B does not reduce peak memory, so these switches have a score of one. For $C_\ell > B$, the score decreases linearly to zero at $C_\ell = B + \Delta$:

$$e_{\mathrm{mem},\ell} = \mathrm{clip}\left(\frac{B + \Delta - C_\ell}{\Delta}, 0, 1\right). \tag{7}$$

The memory-only score does not distinguish between switch positions before break-even point B. The *break-even-cosine* score gives lower values to early switches. When $P < B$, a switch immediately after the prompt, $\tau_\ell = P$, has a score of zero. The score increases to one at B, or at T if the sequence ends before B. After B, the score decreases smoothly over the same range Δ:

$$e_{\cos,\ell} = \begin{cases} \cos^{3/2}\left(\dfrac{\pi}{2}\dfrac{\min(B,T) - \tau_\ell}{\min(B,T) - P}\right), & C_\ell \le B \text{ and } P < \min(B,T), \\ \cos^{3/2}\left(\dfrac{\pi}{2}\,\mathrm{clip}\left(\dfrac{C_\ell - B}{\Delta}, 0, 1\right)\right), & \text{otherwise.} \end{cases} \tag{8}$$

Subtracting P measures progress after the prompt, and $\min(B, T)$ handles sequences that end before B. When $T < B$, never switching has a score of one because the full-attention KV cache remains smaller than the linear-path cache. If $P \geq B$, the first branch is skipped. Just after B, the cosine score decreases more slowly than the memory-only score. Both scores are zero at $B + \Delta$.

Each training run chooses a reward form from Equation 6 and uses the same efficiency-score definition for all layers: $e_\ell = e_{\text{mem},\ell}$ for memory-only or $e_\ell = e_{\cos,\ell}$ for break-even cosine.

4.4 Router Training: GRPO

Algorithm 1 summarizes the training procedure. For rollout k, let $\mathcal{D}_k$ denote the layer-token positions at which a routing-mask value is sampled. This set includes both stay-full values and the final zero for every layer that switches. Let $G_{\max}$ be the maximum generation length.

We use group relative policy optimization (GRPO) (Shao et al., 2024) without a learned value model to train the router parameters θ. Both the full-attention and linear-attention paths of the LLM remain frozen, and the hidden states used in the context-aware part of the router are detached.

Algorithm 1 Training the adaptive router

for prompt–answer pair (x, y) **do**
  for $k = 1, \ldots, K$ **do**
    $\hat{y}_k, \{m_d^{(k)}\}_{d \in \mathcal{D}_k} \sim \text{Generate}(x; \theta)$
    $g_k \leftarrow \sum_{d \in \mathcal{D}_k} \nabla_\theta \log \pi_\theta(m_d^{(k)})$
    Compute R_k using Equation 6
  end for
  $A_k \leftarrow \dfrac{R_k - \frac{1}{K}\sum R_j}{\text{std}(R_j) + \delta}, \quad k = 1, \ldots, K$
  $\bar{g} \leftarrow \dfrac{1}{KLG_{\max}} \sum_{k=1}^{K} A_k g_k$
  $\theta \leftarrow \text{OptimizerStep}(\theta, -\bar{g})$
end for

For each prompt, we sample K responses using different routing decisions and compute their rewards as defined in Section 4.4. The advantage of response k is its reward relative to the other responses for the same prompt:

$$A_k = \frac{R_k - \frac{1}{K}\sum_{j=1}^{K} R_j}{\text{std}(R_j) + \delta}, \tag{9}$$

where δ is a small constant for numerical stability. A positive A_k means that the response performed better than the other responses in its group, while a negative value means that it performed worse.

The router is updated with the log-probability gradient of the sampled routing-mask values:

$$\nabla_\theta \mathcal{L}_{\text{router}} \propto -\frac{1}{KLG_{\max}} \sum_{k=1}^{K} A_k \sum_{d \in \mathcal{D}_k} \nabla_\theta \log \pi_\theta \left(m_d^{(k)} \right). \tag{10}$$

The inner sum includes every sampled mask value, including the zero that switches a layer to linear attention. Thus a positive-advantage response increases the probability of its sampled routing trajectory, while a negative-advantage response decreases it. We use max-completion normalization, dividing by $KLG_{\max}$: the maximum possible number of layer-token decisions in the rollout group. This fixed denominator keeps the overall gradient scale bounded and avoids dividing by the realized number of decisions, which would give each decision in a short or early-switching trajectory greater weight. The normalization does not change A_k: every sampled decision has the same coefficient, so trajectories with more routing decisions can still contribute more total gradient. Each group is generated by the current router and used for one update, so the usual GRPO probability ratio is one and clipping does not change the gradient.

5 Experiments

5.1 Experimental Setup

All routing experiments use the Qwen3-8B full-attention model and its SWA-64+GDN linear path. The linear path was trained on 4.5B OpenThoughts3 tokens, as described in Section 3 and Table 4. In these experiments we only train the router parameters using Algorithm 1, while the full-attention and linear-path parameters remain frozen.

The router training data is drawn from the GuRU (Cheng et al., 2025) dataset. We combine its math and code subsets and filter this mixture to retain only examples that the full Qwen3-8B model

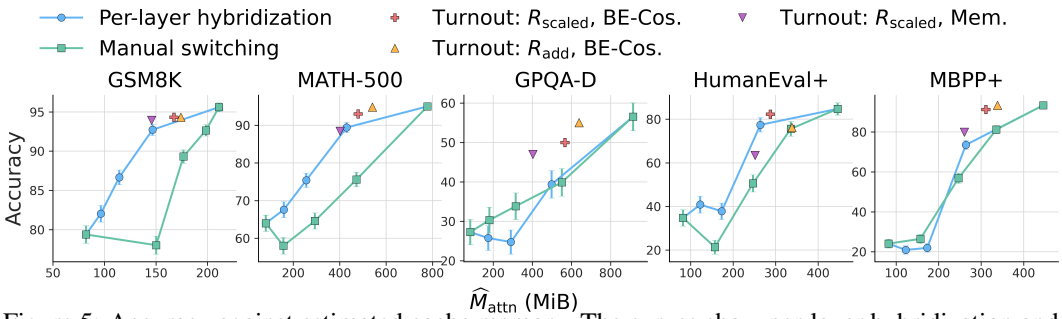


Figure 5: Accuracy against estimated cache memory. The curves show per-layer hybridization and manual switching. The separate markers show TURNOUT routers trained with different reward functions.

answers correctly, ensuring that router training focuses on preserving existing model capabilities while improving memory efficiency. Each router is trained for 100 steps on four GPUs, using a per-GPU batch size of 8 and a group size of $K = 4$. Unless specified otherwise, we set the efficiency weight α to 0.1, the break-even point B to 512 tokens, and the decay parameter Δ to 8192 tokens. We do not tune the optimizer hyperparameters and keep them fixed across experiments. Appendix B.3 and Table 9 provide the complete router-training configuration. Evaluation uses the benchmark protocols described in Section 3.

5.2 ACCURACY-MEMORY TRADE-OFF

We compare different rewards defined in Section 4.4. We use accuracy on reasoning benchmarks to measure the quality of the learned router policy and a cache-memory proxy as the efficiency metric. We calculate the total cache size at each decoding step and use the maximum across all decoding steps as the memory proxy for a given response. For unswitched layers, we assume that key-value states are stored in bfloat16. Switched layers store the fixed linear-attention state in float32 and the sliding-window cache and attention sinks in bfloat16. We then average this proxy over all generated traces in a given benchmark.

As upper and lower bounds, we report the performance of the all-full and all-linear models. We also compare the learned router with two fixed strategies: per-layer hybrid models, in which we keep 10%, 25%, or 50% of the layers in full-attention mode, and manual switching, in which we manually set the switching point to 1k, 2k, or 4k tokens. For each per-layer hybrid model, we select the layers with the largest final activation-distillation losses and use the trained linear path in all other layers. The full-attention layers use the original Qwen3-8B weights, whereas the linear layers use the weights from the all-linear model. We do not retrain these hybrid models. For manual switching, all layers use full attention for the specified number of generated tokens and then switch to the linear path.

In Figure 5, we plot the resulting TURNOUT models and the baselines on the accuracy-memory plane. Early in training, we observed that the additive reward with memory-only efficiency produced a collapsed router that always switched to linear attention immediately. Therefore, we excluded this configuration from the experiment sweep. Figure 5 shows that learned switching improves the accuracy–memory trade-off over manual switching and static per-layer hybridization, especially on GPQA-D and MBPP+. These results suggest that learning which layers retain full attention, and for how long, improves accuracy at comparable cache memory. We report detailed results in Table 8.

Router Analysis Figure 6 compares router learned with three different rewards. The scaled break-even-cosine router shows the strongest task dependence, switching several layers earlier for code and science than for math. The additive variant keeps more layers in full attention, whereas the memory-only variant learns an almost task-independent pattern in which each layer switches either very early or very late. Thus, the reward design strongly affects the routing strategy learned by TURNOUT.

Ablations To study how the reward hyperparameters affect the learned policy, we vary the decay parameter Δ and the efficiency weight α while fixing the objective to the accuracy-scaled reward with

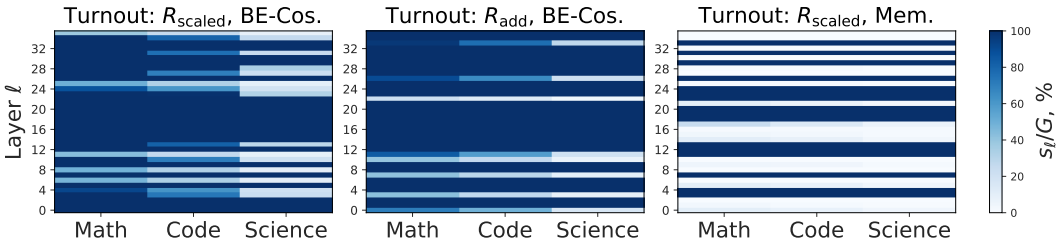


Figure 6: Layer-wise routing strategies for the cosine-scaled, cosine-additive, and memory-scaled routers. Benchmarks are grouped into math, code, and science. Each cell shows the median fraction of the generated response processed with full attention. Darker cells denote later switching.

Table 1: TURNOUT ablations of the reward hyperparameters and context input.

Router config.			Accuracy						Cache saving
Context	Δ	α	GSM8K	GPQA-D	MATH-500	MBPP+	HE+	Avg.	(%) ↑
Full attention			95.6	56.6	95.0	93.4	84.8	85.1	0.0
All linear			79.4	27.3	64.0	24.1	34.8	45.9	69.4
✗	8K	0.1	93.4	45.5	91.4	77.0	**82.9**	78.0	17.4
✓	8K	0.1	**94.3**	50.0	93.0	91.3	82.3	82.2	**25.4**
✓	16K	0.1	94.0	54.0	**94.6**	**93.7**	79.9	**83.2**	10.3
✓	8K	1.0	94.2	46.0	93.0	68.0	76.8	75.6	14.6
✓	16K	1.0	93.6	**59.6**	93.8	91.5	76.2	82.9	14.7

the break-even-cosine efficiency score. We also ablate the context branch by training a schedule-only router that does not receive the hidden-state context input.

Table 1 reports benchmark accuracy and relative cache savings for each configuration. For each sample, we compare the router's peak cache-memory proxy M_{router} with the full-attention KV-cache M_{full} and report the relative saving $1 - M_{\text{router}}/M_{\text{full}}$. The context input improves the accuracy-memory trade-off: it raises average accuracy from 78.0 to 82.2 while increasing cache saving from 17.4% to 25.4% given all other hyperparameter fixed. Increasing Δ to 16K further raises average accuracy to 83.2, only 1.9 points below full attention, but reduces cache saving to 10.3%.

5.3 MEMORY FOOTPRINT AND PARAMETER OVERHEAD

Table 2 summarizes the estimated memory and parameter overhead. The linear path adds 9.09M parameters (0.11% of the original Qwen3-8B), while the RL router adds only 0.65M parameters (0.0079%). Memory estimates include model weights and attention state; TURNOUT uses only MATH-500 routing decisions from the accuracy-scaled break-even-cosine router. At a layer's switching step, we count both its accumulated full-attention KV cache and its linear state before releasing the KV cache. At 16K tokens, TURNOUT reduces the estimated memory by 0.86 GiB per sequence relative to full attention while adding only 0.12% parameters.

Table 2: Memory and parameter overhead.

Method	Added parameters		Memory (GiB)		
	Count	Overhead	Params	Cache (1K)	Cache (16K)
MHA	–	–	15.26	0.14	2.25
Linear	9.09M	0.11%	15.27	0.08	0.08
TURNOUT	9.74M	0.12%	15.28	0.12	1.39

6 CONCLUSION

We introduced TURNOUT, a hybrid post-hoc linearization approach that learns when each layer should switch from full to linear attention during generation. Across reasoning benchmarks, learned routing generally provides a better accuracy–memory trade-off than fixed switching, while our analysis shows that the reward design determines whether the learned policy is task-dependent or largely static.

AI USE STATEMENT

In this work, we used generative AI tools to help draft the text and correct the grammar. We reviewed all AI-assisted work by rereading and checking the text. We also used AI to support qualitative and thematic data analysis by generating plots and tables based on experimental results. We reviewed all generated analysis code and ensured that all plots and figures use the correct experimental data.

We did not use generative AI tools to design or provide feedback on the research methodology or experiments, interpret results, propose or refine hypotheses, helping develop theoretical models or conceptual frameworks, implement methods, or clean and reformat datasets.

Lastly, the following tasks are not applicable to this work: generating synthetic datasets, formulating mathematical claims, providing critical ingredients for proving mathematical claims, assisting with writing proofs, and assisting with translation.

REFERENCES

Aviv Bick, Tobias Katsch, Nimit Sohoni, Arjun Desai, and Albert Gu. Llamba: Scaling distilled recurrent models for efficient language processing. *arXiv preprint arXiv:2502.14458*, 2025. URL https://arxiv.org/abs/2502.14458.

Zhoujun Cheng, Shibo Hao, Tianyang Liu, Fan Zhou, Yutao Xie, Feng Yao, Yuexin Bian, Yonghao Zhuang, Nilabjo Dey, Yuheng Zha, Yi Gu, Kun Zhou, Yuqi Wang, Yuan Li, Richard Fan, Jianshu She, Chengqian Gao, Abulhair Saparov, Haonan Li, Taylor W. Killian, Mikhail Yurochkin, Zhengzhong Liu, Eric P. Xing, and Zhiting Hu. Revisiting reinforcement learning for llm reasoning from a cross-domain perspective, 2025. URL https://arxiv.org/abs/2506.14965.

Krzysztof Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Davis, Afroz Mohiuddin, Lukasz Kaiser, David Belanger, Lucy Colwell, and Adrian Weller. Rethinking attention with performers. *arXiv preprint arXiv:2009.14794*, 2020. URL https://arxiv.org/abs/2009.14794.

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, 2021. URL https://arxiv.org/abs/2110.14168.

Tri Dao. Flashattention-2: Faster attention with better parallelism and work partitioning. *arXiv preprint arXiv:2307.08691*, 2023. URL https://arxiv.org/abs/2307.08691.

Tri Dao and Albert Gu. Transformers are ssms: Generalized models and efficient algorithms through structured state space duality. *arXiv preprint arXiv:2405.21060*, 2024. URL https://arxiv.org/abs/2405.21060.

Evgenii Egorov, Hanno Ackermann, Markus Nagel, and Hong Cai. Myosotis: structured computation for attention like layer. *arXiv preprint arXiv:2509.20503*, 2025. URL https://arxiv.org/abs/2509.20503.

Aaron Grattafiori et al. The llama 3 herd of models, 2024. URL https://arxiv.org/abs/2407.21783.

Yuxian Gu, Qinghao Hu, Haocheng Xi, Junyu Chen, Shang Yang, Song Han, and Han Cai. Jet-Nemotron: Efficient language model with post neural architecture search. In *Advances in Neural Information Processing Systems*, volume 38, pp. 47191–47218, 2025. doi: 10.52202/085713-1572. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/4364fbe773a77fb373df5c7f1d4cbb53-Abstract-Conference.html.

Etash Guha et al. Openthoughts: Data recipes for reasoning models, 2025. URL https://arxiv.org/abs/2506.04178.

Lukas Hauzenberger, Niklas Schmidinger, Thomas Schmied, Anamaria-Roberta Hartl, David Stap, Pieter-Jan Hoedt, Maximilian Beck, Sebastian Böck, Günter Klambauer, and Sepp Hochreiter. Effective distillation to hybrid xlstm architectures. In *Proceedings of the 43rd International Conference on Machine Learning*, 2026. URL https://arxiv.org/abs/2603.15590.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding, 2021a. URL https://arxiv.org/abs/2009.03300.

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, 2021b. URL https://arxiv.org/abs/2103.03874.

Alexia Jolicoeur-Martineau, Rhea Sanjay Sukthanker, Pashmina Cameron, and Emy Gervais. Sliding-window beats linear attention. *arXiv preprint arXiv:2608.28444*, 2026. URL https://arxiv.org/abs/2608.28444.

Kimi Team, Yu Zhang, Zongyu Lin, Xingcheng Yao, Jiaxi Hu, Fanqing Meng, Chengyin Liu, Xin Men, Songlin Yang, Zhiyuan Li, Wentao Li, Enzhe Lu, Weizhou Liu, Yanru Chen, Weixin Xu, Longhui Yu, Yejie Wang, Yu Fan, Longguang Zhong, Enming Yuan, Dehao Zhang, Yizhi Zhang, T. Y. Liu, Haiming Wang, Shengjun Fang, Weiran He, Shaowei Liu, Yiwei Li, Jianlin Su, Jiezhong Qiu, Bo Pang, Junjie Yan, Zhejun Jiang, Weixiao Huang, Bohong Yin, Jiacheng You, Chu Wei, Zhengtao Wang, Chao Hong, Yutian Chen, Guanduo Chen, Yucheng Wang, Huabin Zheng, Feng Wang, Yibo Liu, Mengnan Dong, Zheng Zhang, Siyuan Pan, Wenhao Wu, Yuhao Wu, Longyu Guan, Jiawen Tao, Guohong Fu, Xinran Xu, Yuzhi Wang, Guokun Lai, Yuxin Wu, Xinyu Zhou, Zhilin Yang, and Yulun Du. Kimi linear: An expressive, efficient attention architecture. *arXiv preprint arXiv:2510.26692*, 2025. URL https://arxiv.org/abs/2510.26692.

Anna Kuzina, Paul N. Whatmough, and Babak Ehteshami Bejnordi. The key to going linear: Analysis-driven transformer linearization. *arXiv preprint arXiv:2607.07706*, 2026. URL https://arxiv.org/abs/2607.07706.

Disen Lan, Weigao Sun, Jiaxi Hu, Jusen Du, and Yu Cheng. Liger: Linearizing large language models to gated recurrent structures. *arXiv preprint arXiv:2503.01496*, 2025. URL https://arxiv.org/abs/2503.01496.

Jeffrey Li, Alex Chengyu Fang, Georgios Smyrnis, Maor Ivgi, Matt Jordan, Samir Yitzhak Gadre, Hritik Bansal, Etash Guha, Sedrick Scott Keh, Kushal Arora, Saurabh Garg, Rui Xin, Niklas Muennighoff, Reinhard Heckel, Jean Mercat, Mayee Chen, Suchin Gururangan, Mitchell Wortsman, Alon Albalak, Yonatan Bitton, Marianna Nezhurina, Amro Abbas, Cheng-Yu Hsieh, Dhruba Ghosh, Josh Gardner, Maciej Kilian, Hanlin Zhang, Rulin Shao, Sarah I. Pratt, Sunny Sanyal, Gabriel Ilharco, Giannis Daras, Kalyani Marathe, Aaron Gokaslan, Jieyu Zhang, Khyathi Raghavi Chandu, Thao Nguyen, Igor Vasiljevic, Sham M. Kakade, Shuran Song, Sujay Sanghavi, Fartash Faghri, Sewoong Oh, Luke Zettlemoyer, Kyle Lo, Alaaeldin El-Nouby, Hadi Pouransari, Alexander Toshev, Stephanie Wang, Dirk Groeneveld, Luca Soldaini, Pang Wei Koh, Jenia Jitsev, Thomas Kollar, Alexandros G. Dimakis, Yair Carmon, Achal Dave, Ludwig Schmidt, and Vaishaal Shankar. Datacomp-lm: In search of the next generation of training sets for language models. In *Advances in Neural Information Processing Systems*, volume 37, pp. 14200–14282, 2024. URL https://arxiv.org/abs/2406.11794.

Kevin Y. Li, Asher Trockman, Ananda Theertha Suresh, and Ziteng Sun. Multi-mixer models: Flexible sequence modeling with shared representations. *arXiv preprint arXiv:2605.28769*, 2026a. URL https://arxiv.org/abs/2605.28769.

Yixing Li, Ruobing Xie, Zhen Yang, Xingwu Sun, Shuaipeng Li, Weidong Han, Zhanhui Kang, Yu Cheng, Chengzhong Xu, Di Wang, and Jie Jiang. Transmamba: A sequence-level hybrid transformer-mamba language model. *Proceedings of the AAAI Conference on Artificial Intelligence*, 40(38):31823–31833, 2026b. doi: 10.1609/aaai.v40i38.40451. URL https://doi.org/10.1609/aaai.v40i38.40451.

Bo Liu, Rui Wang, Lemeng Wu, Yihao Feng, Peter Stone, and Qiang Liu. Longhorn: State space models are amortized online learners. *arXiv preprint arXiv:2407.14207*, 2024. URL https://arxiv.org/abs/2407.14207.

Jiawei Liu, Chunqiu Steven Xia, Yuyao Wang, and Lingming Zhang. Is your code generated by ChatGPT really correct? rigorous evaluation of large language models for code generation, 2023. URL https://arxiv.org/abs/2305.01210.

Luke McDermott, Robert W. Heath, Jr., and Rahul Parhi. Lola: Low-rank linear attention with sparse caching. *arXiv preprint arXiv:2505.23666*, 2025. URL https://arxiv.org/abs/2505.23666.

Weikang Meng, Liangyu Huo, Yadan Luo, Jiawen Guan, Jingyi Zhang, Yingjian Li, and Zheng Zhang. Still: Selecting tokens for intra-layer hybrid attention to linearize llms. *arXiv preprint arXiv:2602.02180*, 2026. URL https://arxiv.org/abs/2602.02180.

Jean Mercat, Igor Vasiljevic, Sedrick Keh, Kushal Arora, Achal Dave, Adrien Gaidon, and Thomas Kollar. Linearizing large language models. *arXiv preprint arXiv:2405.06640*, 2024. URL https://arxiv.org/abs/2405.06640.

Chien Van Nguyen, Huy Nguyen, Ruiyi Zhang, Hanieh Deilamsalehy, Puneet Mathur, Viet Dac Lai, Haoliang Wang, Jayakumar Subramanian, Ryan A. Rossi, Trung Bui, Nikos Vlassis, Franck Dernoncourt, and Thien Huu Nguyen. Lizard: An efficient linearization framework for large language models. *arXiv preprint arXiv:2507.09025*, 2025. URL https://arxiv.org/abs/2507.09025.

Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Quan Ngoc Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernández. The LAMBADA dataset: Word prediction requiring a broad discourse context, 2016. URL https://arxiv.org/abs/1606.06031.

David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. GPQA: A graduate-level google-proof q&a benchmark, 2023. URL https://arxiv.org/abs/2311.12022.

Yeonju Ro, Zhenyu Zhang, Souvik Kundu, Zhangyang Wang, and Aditya Akella. On-the-fly adaptive distillation of transformer to dual-state linear attention. *arXiv preprint arXiv:2506.09316*, 2025. URL https://arxiv.org/abs/2506.09316.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Yutao Sun, Li Dong, Shaohan Huang, Shuming Ma, Yuqing Xia, Jilong Xue, Jianyong Wang, and Furu Wei. Retentive network: A successor to transformer for large language models. *arXiv preprint arXiv:2307.08621*, 2023. URL https://arxiv.org/abs/2307.08621.

Junxiong Wang, Daniele Paliotta, Avner May, Alexander M. Rush, and Tri Dao. The mamba in the llama: Distilling and accelerating hybrid models. In *Advances in Neural Information Processing Systems*, volume 37, pp. 62432–62457, 2024. URL https://arxiv.org/abs/2408.15237.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks. *arXiv preprint arXiv:2309.17453*, 2023. URL https://arxiv.org/abs/2309.17453.

An Yang et al. Qwen3 technical report, 2025a. URL https://arxiv.org/abs/2505.09388.

Songlin Yang, Bailin Wang, Yikang Shen, Rameswar Panda, and Yoon Kim. Gated linear attention transformers with hardware-efficient training. *arXiv preprint arXiv:2312.06635*, 2023. URL https://arxiv.org/abs/2312.06635.

Songlin Yang, Bailin Wang, Yu Zhang, Yikang Shen, and Yoon Kim. Parallelizing linear transformers with the delta rule over sequence length. In *Advances in Neural Information Processing Systems*, volume 37, pp. 115491–115522, 2024. URL https://arxiv.org/abs/2406.06484.

Songlin Yang, Jan Kautz, and Ali Hatamizadeh. Gated delta networks: Improving mamba2 with delta rule. In *International Conference on Learning Representations (ICLR)*, 2025b. URL https://arxiv.org/abs/2412.06464.

Michael Zhang, Simran Arora, Rahul Chalamala, Alan Wu, Benjamin Spector, Aaryan Singhal, Krithik Ramesh, and Christopher Ré. Lolcats: On low-rank linearizing of large language models. *arXiv preprint arXiv:2410.10254*, 2024a. URL https://arxiv.org/abs/2410.10254.

Michael Zhang, Kush Bhatia, Hermann Kumbong, and Christopher Ré. The hedgehog & the porcupine: Expressive linear attentions with softmax mimicry. *arXiv preprint arXiv:2402.04347*, 2024b. URL https://arxiv.org/abs/2402.04347.

A LINEARIZATION

A.1 LINEARIZATION TRAINING AND OBJECTIVE ABLATIONS

The linearized model uses the short-convolution and LoRA-on-o_{proj} architecture described in Section 3. Let $x_{1:N}$ be a training sequence, with teacher and student logits z_t^{T} and z_t^{S}. We evaluate next-token prediction, logit and activation distillation loss function.

Next-token prediction is a cross entropy loss

$$\mathcal{L}_{\text{CE}} = -\frac{1}{N-1}\sum_{t=1}^{N-1} \log p_{\text{S}}(x_{t+1} \mid x_{\le t}), \tag{11}$$

Logit distillation uses KL-divergence between teacher and student token distributions. With $p_t^{\text{T}} = \text{softmax}(z_t^{\text{T}}/\tau)$ and $p_t^{\text{S}} = \text{softmax}(z_t^{\text{S}}/\tau)$, the loss is

$$\mathcal{L}_{\text{KL}} = \frac{\tau^2}{N-1}\sum_{t=1}^{N-1} \text{KL}\big(p_t^{\text{T}} \,\|\, p_t^{\text{S}}\big)\,, \tag{12}$$

where τ is a temperature hyperparameter.

For activation distillation (AD) (Zhang et al., 2024a), $h_{\ell,t}^{\text{T}}$ and $h_{\ell,t}^{\text{S}}$ are the teacher and student activations at layer ℓ. We use

$$\mathcal{L}_{\text{AD}} = \frac{1}{LN}\sum_{\ell_1=1}^{L}\sum_{t=1}^{N}\left\|\frac{h_{\ell,t}^{\text{S}} - h_{\ell,t}^{\text{T}}}{\sigma_{\text{global}}}\right\|_2^2, \quad \sigma_{\text{global}} = \text{Std}\big(\{h_{\ell,t}^{\text{T}}\}_{\ell,t}\big)\,. \tag{13}$$

The same normalizing scalar σ_{global} is shared across layers and positions. Teacher forcing uses the teacher hidden state as an input to the student layer; otherwise the student uses its own upstream activation.

The mix of logit and activation distillation we use the weighted sum of two losses:

$$\mathcal{L}_{\text{mix}} = \lambda_{\text{logit}}\mathcal{L}_{\text{KL}} + \lambda_{\text{AD}}\mathcal{L}_{\text{AD}}. \tag{14}$$

We use $\lambda_{\text{logit}} = \lambda_{\text{AD}} = 0.5$ in our experiments.

Table 3 compares next-token prediction, logit distillation, activation distillation, teacher forcing, and LoRA adapter rank. The full model is a reference, not a linearized variant.

Table 3: Objective and LoRA ablation using DCLM-Edu training dataset. The objective ablation uses 0.08B training tokens; the LoRA ablation uses 0.4B training tokens. ✓ marks a loss component that is used. NTP denotes next-token prediction, Logit denotes logit distillation, AD denotes activation distillation, and TF denotes teacher forcing. LAMBADA denotes LAMBADA OpenAI. LAMBADA and MMLU use no chat template or thinking; GSM8K 0-shot and MATH-500 use the chat template with thinking.

Loss components				LoRA rank	Non-thinking			Thinking		
NTP	Logit	AD	TF		LAMBADA	MMLU	Avg.	GSM8K	MATH-500	Avg.
Full model					64.2	74.9	69.6	95.6	95.0	95.3
Objective ablation										
✓				128	**67.0**	70.4	**68.7**	8.0	8.0	8.0
	✓			128	62.9	72.3	67.6	14.7	**15.2**	**15.0**
		✓		128	61.6	72.4	67.0	5.0	5.8	5.4
		✓	✓	128	62.1	70.7	66.4	2.3	1.8	2.1
	✓	✓		128	62.4	**72.6**	67.5	**15.6**	13.8	14.7
Adapter ablation (mixed Logit+AD)										
	✓	✓		–	62.5	**73.2**	**67.9**	8.3	10.6	9.5
	✓	✓		128	**62.7**	72.8	67.8	**30.4**	**20.8**	**25.6**
	✓	✓		256	62.6	73.0	67.8	28.1	**20.8**	24.5

A.2 TRAINING-DATA ABLATION

We compare three training mixtures: DCLM-Edu (DCLM), OpenThoughts3 (OT), and DCLM followed by OT. The combined starts with 1B DCLM-Edu tokens and then continues on OT until reaching the reported total token count. We report accuracies across all the benchmarks in Table 4. All the experiments use a mix of logit and activation distillation objective without teacher forcing. Sliding window size if fixed at 64, 8 sink tokens and LoRA rank 128.

Table 4: Training-data ablation for the linearized model. LAMBADA and MMLU use no chat template or thinking; GSM8K 0-shot and MATH-500 use long-thinking evaluation. GPQA denotes GPQA-Diamond, and HE+ denotes HumanEval+. Each Avg. column is the unweighted mean within its benchmark group. For Llama 3.1-8B, we use one linearized checkpoint trained on 4.5B OpenThoughts tokens.

				Non-thinking			Thinking					
Model (training data)	SWA	Sink	Tok.	LAMBADA	MMLU	Avg.	GSM8K	MATH-500	GPQA	HE+	MBPP+	Avg.
Qwen3-8B												
Full model	–	–	–	64.2	74.9	69.6	95.6	95.0	56.6	84.8	93.4	85.1
SWA-only	64	8	–	62.0	70.9	66.5	1.8	3.0	3.5	0.0	0.0	1.7
TURNOUT (DCLM)	64	8	0.4B	62.7	72.8	67.8	30.4	20.8	16.2	17.1	22.5	21.4
TURNOUT (DCLM)	64	8	1.0B	**63.0**	73.0	68.0	33.7	25.2	16.7	22.0	23.5	24.2
TURNOUT (DCLM)	64	8	2.0B	**63.0**	**73.3**	**68.2**	36.1	26.2	18.7	16.5	23.8	24.3
TURNOUT (OT)	64	8	0.5B	61.1	70.7	65.9	64.3	53.4	23.2	36.0	27.0	40.8
TURNOUT (OT)	64	8	1.0B	62.4	70.0	66.2	69.1	52.8	**28.3**	**36.6**	29.4	43.2
TURNOUT (OT)	64	8	2.0B	60.8	69.2	65.0	73.5	60.6	23.2	34.1	27.2	43.7
TURNOUT (OT)	64	8	4.5B	57.8	67.9	62.9	**79.4**	**64.0**	27.3	34.8	24.1	**45.9**
TURNOUT (DCLM → OT)	64	8	2.0B	62.6	69.9	66.3	71.7	56.2	20.2	29.9	**33.1**	42.2
TURNOUT (DCLM → OT)	64	8	3.0B	62.1	69.1	65.6	74.4	61.2	19.2	29.9	27.5	42.4
Llama 3.1-8B												
Full model	–	–	–	72.2	68.3	70.3	78.2	48.6	23.7	58.5	73.0	56.4
SWA-only	64	8	–	69.4	64.3	66.9	3.0	6.8	4.0	11.0	38.1	12.6
TURNOUT (OT)	64	8	4.5B	67.0	58.8	62.9	51.2	28.0	8.1	25.6	31.2	28.8

A.3 SLIDING-WINDOW SIZE ABLATION

Figure 2 presents the Qwen3-8B and Llama 3.1-8B sliding-window comparisons. We report the values used to produce these figure in Tables 5 and 6. All variants use the evaluation protocols defined in Section 3. For each model, the trained SWA-64 result is the 4.5B-token OpenThoughts checkpoint from the training-data ablation. SWA-128 and SWA-512 evaluate the same checkpoint with a different SWA size.

Table 5: Qwen3-8B sliding-window ablation corresponding to Figure 2(a). LAMBADA and MMLU use no chat template or thinking; GSM8K 0-shot and MATH-500 use long-thinking evaluation. GPQA uses the Qwen-thinking task and final-answer metric. HE+ denotes HumanEval+; HE+ and MBPP+ report pass@1. Each Avg. column is the unweighted mean within its benchmark group.

				Non-thinking			Thinking					
Model (training data)	SWA	Sink	Tok.	LAMBADA	MMLU	Avg.	GSM8K	MATH-500	GPQA	HE+	MBPP+	Avg.
Full model	–	–	–	64.2	74.9	69.6	95.6	95.0	56.6	84.8	93.4	85.1
SWA-only	64	8	–	62.0	70.9	66.5	1.8	3.0	3.5	0.0	0.0	1.7
SWA-only	128	8	–	64.1	74.1	69.1	9.2	9.0	3.0	0.0	0.3	4.3
SWA-only	512	8	–	64.1	74.9	69.5	74.1	47.4	12.6	6.7	8.7	29.9
TURNOUT (OT)	64	8	4.5B	57.8	67.9	62.9	79.4	64.0	27.3	34.8	24.1	45.9
TURNOUT (OT)	128	8	4.5B	59.1	69.6	64.4	83.9	68.4	27.8	37.8	28.0	49.2
TURNOUT (OT)	512	8	4.5B	59.1	69.8	64.5	90.7	76.8	28.3	48.8	50.8	59.1

Table 6: Llama 3.1-8B sliding-window ablation corresponding to Figure 2(b). LAMBADA and MMLU use no chat template or thinking; GSM8K 0-shot and MATH-500 use long-thinking evaluation. GPQA denotes GPQA-Diamond, and HE+ denotes HumanEval+. HE+ and MBPP+ report pass@1. Each Avg. column is the unweighted mean within its benchmark group.

				Non-thinking			Thinking					
Model (training data)	SWA	Sink	Tok.	LAMBADA	MMLU	Avg.	GSM8K	MATH-500	GPQA	HE+	MBPP+	Avg.
Full model	–	–	–	72.2	68.3	70.3	78.2	48.6	23.7	58.5	73.0	56.4
SWA-only	64	8	–	69.4	64.3	66.9	3.0	6.8	4.0	11.0	38.1	12.6
SWA-only	128	8	–	72.2	66.5	69.4	27.7	18.4	3.0	31.1	65.6	29.2
SWA-only	512	8	–	72.2	68.1	70.2	78.5	49.0	16.7	57.9	74.9	55.4
TURNOUT (OT)	64	8	4.5B	67.0	58.8	62.9	51.2	28.0	8.1	25.6	31.2	28.8
TURNOUT (OT)	128	8	4.5B	68.8	60.1	64.5	65.0	36.8	9.6	36.0	45.2	38.5
TURNOUT (OT)	512	8	4.5B	68.8	61.3	65.1	72.4	44.2	16.7	45.7	62.2	48.2

A.4 LINEARIZATION HYPERPARAMETERS

Table 7: Linearization training and evaluation hyperparameters used for both Qwen3-8B and Llama 3.1-8B.

Hyperparameter	Value
Training data	
Teacher / backbone f_{T}	`Qwen/Qwen3-8B`;`Meta-Llama-3.1-8B-Instruct`
Dataset $\mathcal{D}$	OpenThoughts3-1.2M (train)
Input formatting	Each model's chat template
Tokens used in the loss	Assistant-response tokens only
Gradient propagation length C	4096 tokens
Input-token budget	4.5B (0.25 epoch)
Linear path	
Linear mixer	GDN in every attention layer
SWA window W	64
Sink tokens s	8
Gate dimensions	Scalar key gate; scalar β
β activation	Softplus
Gate initialization	$b_g = 1$, $b_\beta = -1.5$
Short convolution	$k = 8$, bias $=$ True
Trainable parameters	GDN, short convolution, and LoRA; backbone frozen
LoRA	
Target module	o_{proj}
Rank r	128
Scale α	8
Dropout p	0
Objective	
Loss weights $(\lambda_{\mathrm{KL}}, \lambda_{\mathrm{AD}})$	$(0.5, 0.5)$
KL direction	$\mathrm{KL}(p_{\mathrm{T}} \,\|\, p_{\mathrm{S}})$
Distillation temperature τ	2
Activation loss	Squared ℓ_2
Teacher forcing	False
Optimization	
Optimizer	AdamW
Learning rate η	10^{-3}
Adam (β_1, β_2)	$(0.9, 0.95)$
Adam ϵ	10^{-8}
Weight decay	0
LR schedule	Cosine
Warmup steps	100
Gradient clipping	1
Training steps S	150,000
Batch size per GPU	1
Number of GPUs	2
Gradient accumulation	1
Precision	bfloat16
Seed	1
Generation (reasoning and code)	
Sampling	True
Temperature T_{gen}	0.6
Top-p	0.95
Top-k	20
Maximum generated tokens $G_{\max}$	16,384
Samples per prompt	1

B Reinforcement Learning

B.1 Router Architecture

Schedule Let $\boldsymbol{\psi}(t)$ be the log-spaced sinusoidal encoding of position t, with F frequencies and maximum period Ω:

$$\boldsymbol{\psi}(t) = \left[\sin\left(t\Omega^{-i/(F-1)}\right); \cos\left(t\Omega^{-i/(F-1)}\right)\right]_{i=0}^{F-1}, \qquad \Omega = 10{,}000. \tag{15}$$

The positional input in Section 4.2 is

$$\mathbf{x}_t^{\text{schedule}} = \left[\boldsymbol{\psi}(t); \frac{t}{B}; \text{clip}\left(\frac{t}{B}, 0, 1\right); \max\left(\frac{t-B}{B}, 0\right); \mathbf{m}_{t-1}\right], \qquad \mathbf{m}_{t-1} \in \{0,1\}^L. \tag{16}$$

This vector has $2F + 3 + L$ entries. We use a per-layer MLP to produce the schedule-dependent part of the router:

$$\begin{aligned} \nu_{\ell,t}^{\text{schedule}} &= W_\ell^{\text{out}} \,\text{SiLU}\big(W_\ell^{\text{in}} \mathbf{x}_t^{\text{schedule}} + \mathbf{b}_\ell^{\text{in}}\big) + b_\ell^{\text{out}}. \\ W_\ell^{\text{in}} &\in \mathbb{R}^{H^{\text{schedule}} \times (2F+3+L)}, \quad \mathbf{b}_\ell^{\text{in}} \in \mathbb{R}^{H^{\text{schedule}}}, \quad W_\ell^{\text{out}} \in \mathbb{R}^{1 \times H^{\text{schedule}}}, \quad b_\ell^{\text{out}} \in \mathbb{R}. \end{aligned} \tag{17}$$

We initialize W_ℓ^{in} with Xavier uniform, set $\mathbf{b}_\ell^{\text{in}} = 0$ and $W_\ell^{\text{out}} = 0$, and set $b_\ell^{\text{out}} = 6$. This means that, at initialization, the router has a stay-full probability of approximately 99.8%, and each layer switches, on average, at token 500.

Context The context input is the hidden representation of the corresponding attention layer:

$$\text{sg}[\mathbf{h}_{\ell,t}] \in \mathbb{R}^{H^{\text{model}}}.$$

Here H^{model} is the hidden size of the frozen language model. Because it is typically large, the first layer of the context MLP is shared across the LLM layers to save router parameters. Only its readout is layer-specific, so it carries the index ℓ:

$$\begin{aligned} \nu_{\ell,t}^{\text{context}} &= W_\ell \,\text{SiLU}\left(W^{\text{c}} \,\text{sg}[\mathbf{h}_{\ell,t}] + \mathbf{b}^{\text{c}}\right). \\ W^{\text{c}} &\in \mathbb{R}^{H^{\text{context}} \times H^{\text{model}}}, \quad \mathbf{b}^{\text{c}} \in \mathbb{R}^{H^{\text{context}}}, \quad W_\ell \in \mathbb{R}^{1 \times H^{\text{context}}}. \end{aligned} \tag{18}$$

We initialize W^{c} with Xavier uniform, set $\mathbf{b}^{\text{c}} = 0$, and set $W_\ell = 0$.

The total score is $\nu_{\ell,t} = \nu_{\ell,t}^{\text{schedule}} + \nu_{\ell,t}^{\text{context}}$.

Table 9 specifies the corresponding architecture hyperparameters.

B.2 ACCURACY–MEMORY RESULTS

Table 8 reports the numerical values used to prodce Figure 5.

Table 8: Accuracy and relative cache saving for the configurations in Figure 5.

Router configuration					Accuracy						Cache saving
Reward shape	Efficiency score	Context	Δ	α	GSM8K	GPQA-D	MATH-500	MBPP+	HE+	Avg.	(%) ↑
Full attention					95.6	56.6	95.0	93.4	84.8	85.1	0.0
All linear					79.4	27.3	64.0	24.1	34.8	45.9	69.4
Per-layer hybridization (10% full layers)					82.0	25.8	67.6	20.9	40.9	47.4	61.7
Per-layer hybridization (25% full layers)					86.7	24.7	75.4	22.0	37.8	49.3	52.1
Per-layer hybridization (50% full layers)					92.7	39.4	89.4	73.5	77.4	74.5	34.7
Manual switching (1K tokens)					78.0	30.3	58.0	26.5	21.3	42.8	50.8
Manual switching (2K tokens)					89.3	33.8	64.6	56.9	50.6	59.0	27.9
Manual switching (4K tokens)					92.6	39.9	75.6	81.2	75.6	73.0	12.0
Acc.-scaled	Break-even cos.	✓	8K	0.1	94.3	50.0	93.0	91.3	82.3	82.2	25.4
Additive	Break-even cos.	✓	8K	0.1	94.3	55.1	94.8	93.4	76.2	82.8	14.4
Acc.-scaled	Memory-only	✓	8K	0.1	93.9	47.0	88.4	79.9	63.4	74.5	39.3

B.3 Router Training Hyperparameters

Table 9: Router-training hyperparameters for Table 1.

Hyperparameter	Value
Router	
Schedule hidden size H^{schedule}	32
Schedule activation	SiLU
Sinusoidal frequencies F	32
Sinusoidal maximum period Ω	10,000
Context hidden size H^{context}	128
Context activation	SiLU
Reward	
Efficiency weight α	0.1
Break-even point B	512 tokens
Decay Δ	8192 tokens
Optimization	
Optimizer	AdamW
Learning rate η	10^{-2}
Adam (β_1, β_2)	(0.9, 0.999)
Adam ϵ	10^{-8}
Weight decay	0
LR schedule	Constant
Warmup steps	0
Gradient clipping	1
Training steps	100
Group size K	4
Batch size per GPU	8
Number of GPUs	4
Gradient accumulation	1
Advantage constant δ	10^{-6}
Precision	bfloat16
Seed	0
Generation	
Sampling	True
Temperature	1.0
Top-p	–
Top-k	–
Maximum generated tokens	16,384