Switching Linear Attention

Expressive retrieval with a fixed-size recurrent state

Paper by Hyun Dong Lee, Xavier Gonzalez, Nicolas Zucchet,
E. Kelly Buchanan, Emily B. Fox, and Scott W. Linderman

arXiv:2609.39034v1 · Research group discussion, 28 October 2026

Published in COLM 2026 pre-proceedings (venue DOI pending) · Official code

SwiLA changes the function stored in recurrent memory

One-sentence takeaway

Replace one online linear regressor with a mixture of linear regressors: input- and error-dependent updates, query-dependent readout, fixed state size.

What improves

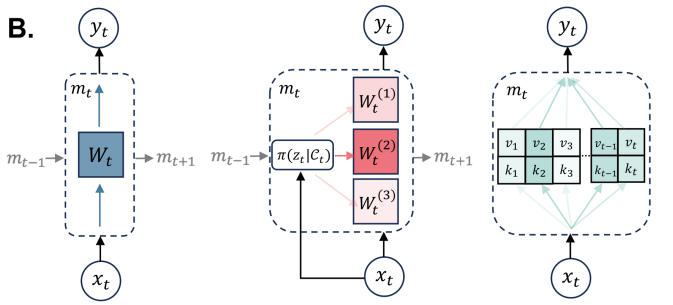
- Context-dependent associative recall
- Approximation of softmax retrieval at matched recurrent state size

Paper, abstract and Secs. 1, 3, 5–6; PDF pp. 1–10.

What costs more

- Nonlinear, sequential state update
- More training memory and lower throughput than GDN

The bottleneck is the query-to-value map



- Softmax attention retains all past KV pairs; cache grows with sequence length.
- Linear attention compresses history into one matrix per head.
- SwiLA stores several matrices and mixes their predictions at readout.

Original paper Fig. 1B, PDF p. 3; image cropped from the authors' PDF figure.

Test-time regression turns attention into online memorization

$$m_t \approx \arg\min_{m \in \mathcal{M}} \sum_{i=1}^{t} \|v_i - m(k_i)\|_2^2, \qquad o_t = m_t(q_t).$$

- Keys k_i, values v_i, and queries q_t are input projections.
- The recurrence updates the *fast weights* of m_t inside each sequence.
- DeltaNet: one linear map, $m_t(q) = W_t q$.
- Softmax: a nonparametric kernel smoother over stored KV pairs.

SwiLA design choice

Function class: mixture of linear regressors. Update: online EM-style stochastic gradient step.

Paper Sec. 2, Eqs. (2.1)–(2.2), PDF p. 3; Sec. 3, PDF p. 3.

Each output dimension can choose its own expert

$$p(z_{td} = j \mid k_t) = \pi_{jd}(k_t), \qquad v_{td} \mid z_{td} = j, k_t \sim \mathcal{N}(w_{jd}^\top k_t, 1).$$

- $j = 1, \ldots, J$: expert; $d = 1, \ldots, D$: value dimension.
- $w_{jd} \in \mathbb{R}^D$; all experts form $W \in \mathbb{R}^{J \times D \times D}$.
- Independent dimension-wise assignments permit J^D combinations, while the matrix state has JD^2 scalars per head.

Interpretation

This is a soft, dimension-wise switch among linear retrieval maps.

Paper Sec. 3.1, Eq. (3.1), PDF p. 3. J^D counts possible assignments, not guaranteed realized modes.

Write: responsibility weights the delta-rule error

$$\delta_{tjd} = v_{td} - w_{t-1,jd}^{\top} k_t, \qquad r_{tjd} = \frac{\pi_{jd}(k_t) e^{-\delta_{tjd}^2/2}}{\sum_{j'} \pi_{j'd}(k_t) e^{-\delta_{tj'd}^2/2}}.$$

$$\boxed{w_{tjd} = w_{t-1,jd} + \beta_{tjd} r_{tjd} \delta_{tjd} k_t.}$$

- Prior π: input-based routing; r: posterior after observing v_{td}.
- δ: prediction error; $\beta \in [0, 1]$: learned step size.
- r depends on the old state, making the recurrence nonlinear.

Paper Sec. 3.2, Eqs. (3.2)–(3.4), PDF p. 4; gradient derivation App. B.1, p. 20.

Read: the query prior mixes expert predictions

$$o_{td} = \sum_{j=1}^{J} \pi_{jd}(q_t)\, w_{tjd}^{\top} q_t.$$

- The readout uses $\pi(q_t)$, because the value v_{td} is unknown at retrieval.
- A weighted sum of linear maps becomes nonlinear in the input when the weights change with the query.
- The state is still fixed-size with respect to context length: $O(HJD^2)$ matrix scalars for H heads.

Key distinction

Write routing uses observed prediction error; read routing uses the query-side prior.

Paper Sec. 3.2, Eq. (3.5), PDF p. 4; App. B.2, p. 20.

Temporal SwiLA carries context across tokens

$$\bar{r}_{tjd} = (1 - g^k_{td}) r_{t-1,jd} + g^k_{td} \pi^k_{jd}(k_t)$$
$$\bar{\rho}_{tjd} = (1 - g^q_{td}) \bar{\rho}_{t-1,jd} + g^q_{td} \pi^q_{jd}(q_t).$$

- The filtered key distribution becomes the next write prior.
- The filtered query distribution becomes the read prior.
- Gate g near zero preserves context; near one accepts current input.

Why it matters

A context marker may appear once, many positions before the answer.

Paper Sec. 3.3, PDF p. 4; App. B.3, Eqs. (B.3)–(B.5), PDF pp. 20–21.

Gating and balanced routing solve different problems

State decay (the implemented pre-decay form)

$$w_{tjd} = \alpha_{tjd} w_{t-1,jd} + \beta_{tjd} r^{-}_{tjd} \delta^{-}_{tjd} k_t, \quad \delta^{-}_{tjd} = v_{td} - \alpha_{tjd} w^{\top}_{t-1,jd} k_t.$$

Avoiding expert collapse

$$\mathcal{L}_{\mathrm{MI}} = -\lambda_{\mathrm{marg}} H_{\mathrm{marg}} + \lambda_{\mathrm{cond}} H_{\mathrm{cond}}.$$

- High marginal entropy spreads usage across experts.
- Low conditional entropy encourages token-level specialization.
- Training-time noisy router logits are another option; the 374M language models disable that noise.

Paper Secs. 3.4–3.5, Eqs. (3.8)–(3.10), PDF p. 5; App. E.3, p. 30.

SwiLA differs from DeltaNet and input-only memory routing

Layer	Stored state	What controls the update?
DeltaNet	One linear matrix per head	Prediction error, single map
MoM	Several memory matrices	Router based on input, independent of old state
SwiLA	Several regressors, each output dimension routed	Posterior uses input prior *and* old-state prediction error
Softmax attention	All earlier KV pairs	Query–key similarities at read time

Mechanistic claim

SwiLA's state-dependent responsibility creates a nonlinear recurrence; its query-side mixture creates a nonlinear retrieval map.

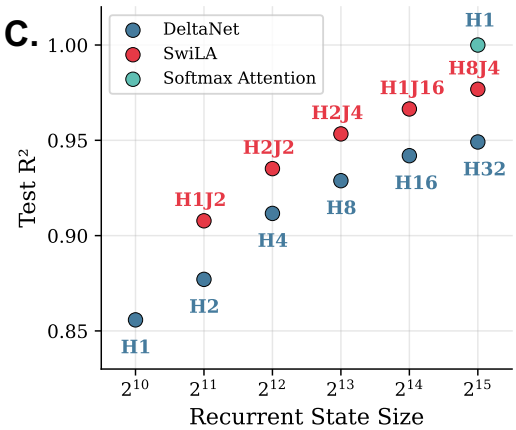
Paper Sec. 4, PDF pp. 5–6.

The experiments test expressivity, recall, and systems cost

Setting	Main question	Primary metric
Softmax fit	Better approximation at equal state size?	Test R^2 ↑
CDAR	Store conflicting key–value maps and persist context?	Answer accuracy ↑
RegBench	Learn a finite automaton from context examples?	Test accuracy ↑
374M language models	Does retrieval improve after 15B training tokens?	Commonsense / recall ↑
1.3B profiling	What is the compute cost?	Throughput and memory
115M ablation	When does expert routing collapse?	Test perplexity

Paper Sec. 5, PDF pp. 6–10; experimental details App. E, pp. 27–33.

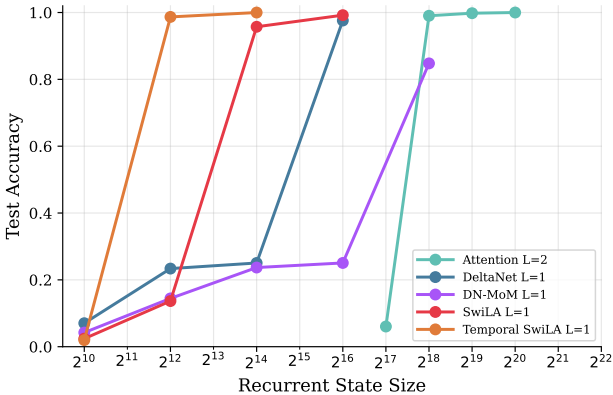
Mixtures better fit a softmax head at matched state size



- Target: causal softmax outputs from clustered inputs, $D = 32$.
- Budget $B = HJ$; each head has a $D \times D$ state.
- At $B = 2$, $H1J2$ exceeds DeltaNet $H2$ in test R^2.
- At $B = 4$, the best split is $H2J2$.

Original Fig. 1C, PDF p. 3; Sec. 5.1, p. 7. $B = 1$ has no multi-expert comparator.

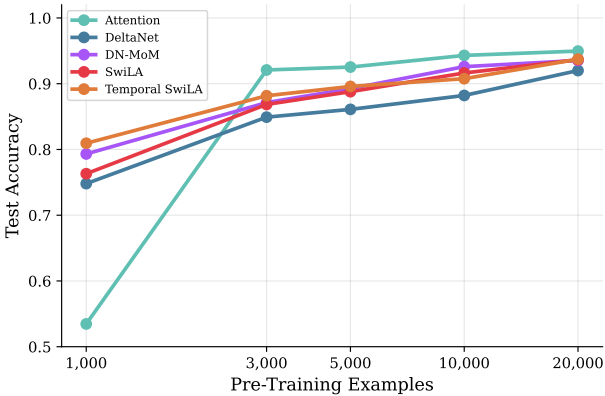
Temporal SwiLA solves context-dependent recall with less state



- Four contexts share the same 32 keys; each context assigns different values.
- Context marker appears once per query block.
- One-layer Temporal SwiLA with $J = 4$ is near-perfect at 2^{12} state scalars.
- One-layer softmax does not solve this setup; two layers do.

Original Fig. 2A, Sec. 5.2, PDF p. 8. Main plot uses best result across learning rates and three seeds; mean results in Fig. S2, p. 28.

RegBench shows a low-data advantage, then convergence



Original Fig. 2B, Sec. 5.2, PDF p. 8; protocol in App. E.2, p. 29.

- Eight-layer models infer a probabilistic finite automaton in context.
- At 1,000 training automata, Temporal SwiLA and DN-MoM exceed Transformer accuracy.
- At 20,000, Transformer reaches the highest accuracy; models are close.

The language-model comparison matches state and parameter scale

- 13 models: 24 layers, width 1,024, about 374M parameters.
- FineWeb-Edu pretraining: 15B tokens, sequence length 4,096.
- SwiLA: 8 heads $\times$ 4 experts $\times$ $64^2 = 131{,}072$ matrix-state scalars per layer.
- DeltaNet, GDN, KDA: $8 \times 128^2 = 131{,}072$.
- Hybrid models place six full-attention layers among 18 recurrent layers.

Evaluation

Table 1: eight commonsense accuracies and perplexities.
Table 2: six recall tasks, contains metric, inputs truncated to 2K tokens.

Paper Sec. 5.3, Tables 1–2, PDF pp. 8–10; App. E.3, p. 30.

KDA leads commonsense; Gated SwiLA is close

Model (374M)	WikiText PPL ↓	Commonsense avg. % ↑
Transformer++	28.2	45.3
DeltaNet	29.8	43.7
GDN	27.7	45.0
KDA	**23.4**	**46.6**
SwiLA	28.8	44.7
Temporal SwiLA	28.6	45.2
Gated SwiLA	27.8	45.4
Gated Temporal SwiLA	27.8	45.4

Gated SwiLA exceeds Transformer++ by 0.1 point; KDA remains strongest.

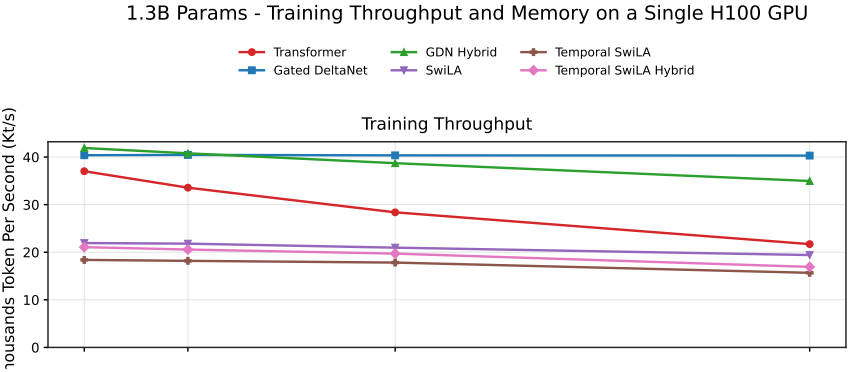
Selected exact values from original Table 1, PDF p. 9. PPL lower is better; accuracy higher is better.

SwiLA improves fixed-state recall; full KV caches still lead

Model (374M)	Recall avg. contains % ↑	SWDE / FDA %
DeltaNet	17.3	7.9 / 5.4
GDN	19.2	9.1 / 5.5
KDA	20.3	13.9 / 5.5
SwiLA	19.7	12.9 / 8.0
Gated SwiLA	20.5	10.2 / 10.1
Gated Temporal SwiLA	**20.6**	12.0 / 7.5
Hybrid GDN	29.5	25.0 / 38.2
Hybrid Temporal SwiLA	31.8	32.3 / 49.1
Transformer++	28.3	28.4 / 29.5
FoX	**33.7**	36.4 / 53.0

Selected exact values from original Table 2, PDF p. 10. All inputs truncated to 2K tokens; hybrid models include six full-attention layers.

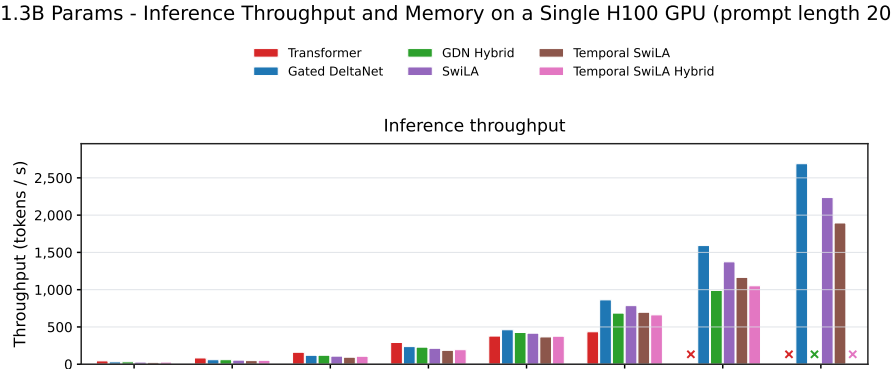
SwiLA's sequential update halves training throughput vs GDN



1.3B models, one H100, 131,072 tokens/update: SwiLA 19–22K tokens/s vs GDN about 40K tokens/s. SwiLA also allocates more peak memory.

Original Fig. S3, PDF p. 31; top panel cropped. Timed update includes forward, backward, clipping, and AdamW step.

Fixed-state inference permits larger batches on one H100



2,048-token prompt + 128-token decode. Batch groups left to right: 1, 2, 4, 8, 16, 32, 64, 128. Transformer OOMs at large batches; SwiLA reaches 128, but GDN remains faster.

Original Fig. S4, PDF p. 32; throughput panel cropped. The paper's 4–5× advantage over Transformer reflects feasible batch sizes, not matched-batch latency.

Temporal persistence or load balancing prevents collapse

115M model, 5B training tokens	Experts	Test PPL ↓
SwiLA, no load balancing	4	36.769
SwiLA, load balancing	4	16.807
Temporal SwiLA, no load balancing	4	16.718
Temporal SwiLA, load balancing	4	**16.715**
Temporal SwiLA, load balancing	16	16.862

- Without balancing, plain SwiLA's KV-side dead-expert fraction exceeds 40%.
- At fixed state size, more experts shrink each expert's projection; 16 experts slightly worsen PPL.

Exact legend values and router diagnostics from original Fig. S5, PDF p. 33; Sec. 5.5, pp. 9–10.

What the paper establishes and what remains expensive

1. A mixture of online regressors yields a nonlinear query readout with a state independent of context length.
2. Temporal routing is especially useful when a context marker must persist across many tokens.
3. In 374M models, pure recurrent SwiLA variants improve recall over DeltaNet; FoX's full KV cache still leads on recall.
4. The nonlinear recurrence currently trains sequentially and costs about $2\times$ GDN training time in the 1.3B profile.

Open systems question

Can chunkwise parallel Newton iterations outperform the current sequential kernel in realistic training?

Paper Sec. 6, PDF p. 10; App. D, pp. 23–26; App. E.4, pp. 31–32.

Paper, venue, and code

- Hyun Dong Lee, Xavier Gonzalez, Nicolas Zucchet, E. Kelly Buchanan, Emily B. Fox, Scott W. Linderman. *Switching Linear Attention*. arXiv:2609.39034v1.
- COLM 2026 pre-proceedings and official accepted-papers list. Conference DOI pending as of 28 October 2026.
- Official implementation.

All figures shown are from the authors' arXiv v1 source package; selected panels are accurately cropped.

Backup: the update is one marginal-likelihood gradient step

$$\ell_{td} = \log \sum_j \pi_{jd}(k_t) \exp\left[-\frac{1}{2}(v_{td} - w_{t-1,jd}^\top k_t)^2\right] + c.$$

$$\nabla_{w_{t-1,jd}} \ell_{td} = r_{tjd}(v_{td} - w_{t-1,jd}^\top k_t) k_t.$$

- EM language highlights latent responsibilities.
- The actual online parameter update uses one stochastic gradient step; no exact batch M-step is performed over stored history.

Paper Eqs. (3.2)–(3.4), PDF p. 4; derivation App. B.1, p. 20.

Backup: proposed parallelization has not sped up training

$$\frac{\partial f}{\partial w_{t-1,jd}} \approx I - g_{tjd} k_t k_t^\top, \quad g_{tjd} = \beta_{tjd} r_{tjd}[1 - \delta_{tjd}^2(1 - r_{tjd})].$$

- Each Newton iteration becomes a linear recurrence with DeltaNet's identity-minus-rank-one structure.
- A chunkwise algorithm could avoid materializing every matrix state.
- The Jacobian ignores cross-expert coupling from responsibility normalization.
- Around 10 iterations converged on synthetic inputs, but the present Triton implementation was no faster than the sequential kernel.

Paper App. D, Eqs. (D.4)–(D.8), PDF pp. 24–25; limitations in Sec. 6, p. 10.