Enhancing Linear Attention with Residual Learning

RLA and Residual Delta Net: remembering prediction errors

Paper by Xunhao Lai, Jialiang Kang, Jianqiao Lu,
Tong Lin, and Pengyu Zhao

arXiv:2509.25223v1 | Research group reading | 13 November 2026

Reading presentation prepared with Codex.

A second recurrent state learns how the first one errs

One-sentence idea

Fit historical prediction residuals with an auxiliary memory, then use it to correct the base memory's output.

- RLA: additive updates; RDN: delta updates for both states.
- RDN WikiText ppl: 16.57, versus GDN's 17.27.
- RLA NIAH: 83.6%, versus sGLA's 76.6%.

Published at EMNLP 2026, main proceedings.
No public implementation found as of 13 November 2026.
This reading uses arXiv v1 (24 September 2025) throughout.

Paper, Secs. 3–4, PDF pp. 4–7, Tables 2–3. Publication/code checked 2026-11-13.

Linear attention is an online key-to-value mapping

$$q_t, k_t \in \mathbb{R}^{d_k}, \quad v_t, o_t \in \mathbb{R}^{d_v}, \quad S_t \in \mathbb{R}^{d_v \times d_k}$$

$$S_t = S_{t-1} + v_t k_t^\top, \qquad o_t = S_t q_t$$

- Historical associations share a fixed-size matrix.
- Linear scaling with sequence length; fixed recurrent state during decoding.
- Interference and limited capacity can impair exact recall.

Design question

Can another simple memory learn the correction that the base memory needs?

Paper, Secs. 1–2.1, PDF pp. 1–2. Column-vector convention follows the main text.

Prediction–correction separates history from the local write

$$o_t = \underbrace{S_{t-1} q_t}_{\text{base prediction}} + \underbrace{v_t (k_t^\top q_t)}_{\text{current correction}}$$

For Gated DeltaNet:

$$o_t = \alpha_t S_{t-1} q_t + \beta_t (v_t - \alpha_t S_{t-1} k_t)(k_t^\top q_t).$$

- The local correction is generated by the current update.
- Delta-rule residuals already depend on historical state.
- RLA makes the correction itself recurrent, retaining earlier errors.

Paper, Sec. 2.3 and Table 1, PDF p. 3. "Single-token" refers to the local update.

Online losses give simple, reusable state updates

Negative inner product gives additive writing

$$\mathcal{L}_{\text{inner}} = -\langle Sk, v \rangle \quad \Longrightarrow \quad S_t = S_{t-1} + v_t k_t^\top$$

Squared error gives the delta rule

$$\mathcal{L}_{\text{inner}} = \tfrac{1}{2} \| Sk - v \|_2^2$$

$$S_t = S_{t-1} + \beta_t (v_t - S_{t-1} k_t) k_t^\top$$

- The state acts as online parameters of a key-to-value predictor.
- Residual fitting adds a second predictor with analogous update rules.

Paper, Sec. 2.2, PDF pp. 2–3; App. A, PDF p. 13.

L2 prediction error provides the auxiliary target

$$\hat{v}_t = S_{t-1} k_t, \qquad r_t = v_t - \hat{v}_t$$

Minimal residual fitting, without gates or clipping

$$\begin{aligned} R_t &= R_{t-1} + r_t k_t^\top, \\ o_t &= S_{t-1} q_t + R_t q_t, \\ S_t &= S_{t-1} + v_t k_t^\top. \end{aligned}$$

- R_t has the same $d_v \times d_k$ shape as S_t.
- The target is an error vector; the auxiliary state persists across tokens.
- The Taylor/Newton argument motivates r_t; no Hessian inversion is implemented.

Paper, Sec. 3.1, PDF pp. 3–4.

RLA decouples base learning from residual correction

$$
\begin{aligned}
r_t &= \mathrm{Clip}_{[-c,c]}(v_t - S_{t-1}k_t), \\
R_t &= \alpha_t R_{t-1} + \gamma_t r_t k_t^\top, \\
S_t &= \alpha_t S_{t-1} + \beta_t v_t k_t^\top, \\
o_t &= \alpha_t S_{t-1} q_t + \gamma_t R_t q_t.
\end{aligned}
$$

- α_t: retention; β_t: base update; γ_t: correction update/readout.
- Residuals read the old, undecayed S_{t-1}.
- The output reads old S_{t-1} and updated R_t.
- Both the write and the readout use γ_t; main experiments use $c = 1$.

Paper, Secs. 3.2–3.4, PDF pp. 4–5; App. D, Listing 1, PDF p. 15.

The correction attends to a history of errors

For $R_0 = 0$, unrolling RLA gives

$$R_t = \sum_{i=1}^{t} \left(\prod_{j=i+1}^{t} \alpha_j \right) \gamma_i r_i k_i^\top.$$

$$\gamma_t R_t q_t = \sum_{i=1}^{t} \gamma_t \gamma_i \left(\prod_{j=i+1}^{t} \alpha_j \right) r_i (k_i^\top q_t).$$

- Similar historical keys retrieve the errors associated with them.
- Decay controls how long an error remains useful.
- The current correction is scaled by γ_t^2 after write and readout.

Teaching expansion of the RLA equations, Sec. 3.4, PDF p. 5; not a separate paper theorem.

RDN uses delta rules for both memory states

Keep the same clipped target $r_t = \mathrm{Clip}_{[-c,c]}(v_t - S_{t-1}k_t)$:

$$
\begin{aligned}
R_t &= \alpha_t R_{t-1}(I - \gamma_t k_t k_t^\top) + \gamma_t r_t k_t^\top, \\
S_t &= \alpha_t S_{t-1}(I - \beta_t k_t k_t^\top) + \beta_t v_t k_t^\top, \\
o_t &= \alpha_t S_{t-1} q_t + \gamma_t R_t q_t.
\end{aligned}
$$

- Base delta update fits v_t; auxiliary delta update fits r_t.
- Auxiliary update internally corrects $r_t - \alpha_t R_{t-1}k_t$.
- This internal error differs from the base residual used to form r_t.
- With normalized keys, the transition shrinks the current key direction.

Paper, Sec. 3.4, PDF p. 5.

Clipping isolates a robust objective inside the target

For scalar error $e = v - \hat{v}$, Huber loss is

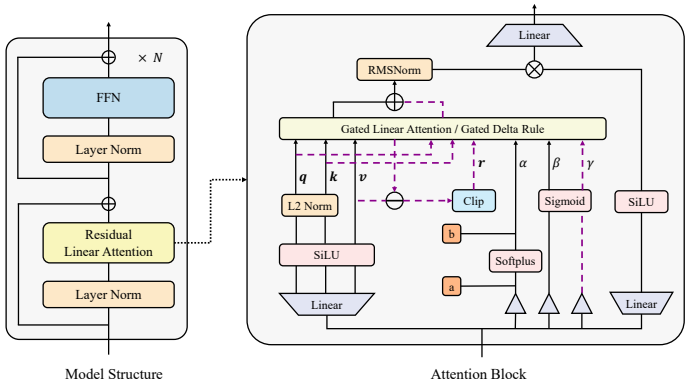
$$\ell_c(e) = \begin{cases} \frac{1}{2}e^2, & |e| \le c, \\ c(|e| - \frac{1}{2}c), & |e| > c. \end{cases}$$

$$-\nabla_{\hat{v}} \mathcal{L}_{\text{Huber}} = \text{Clip}_{[-c,c]}(v - \hat{v}).$$

- Outer mapping loss defines a pseudo-residual; inner loss updates each state.
- Nonlinearity stays in the target; state updates remain kernel-friendly.
- Log-Cosh would yield $\tanh(v - \hat{v})$; the paper proposes this without experiments.

Paper, App. A–B, PDF pp. 13–14. App. B's two subtractive writes conflict with the main text and pseudocode; see backup.

The residual path sits inside the attention block



- Purple path: predict with old state, clip the error, and fit auxiliary memory.
- SiLU and L2-normalized q, k; scalar retention/write/correction gates.

Original Fig. 1, PDF p. 5. Gate definitions and module structure in Sec. 3.4.

Two fitting stages retain linear scaling, with more state

1. Compute the base recurrence and old-state reads $S_{t-1}k_t$.
2. Form clipped residual values r_t.
3. Reuse a linear-attention/delta kernel to fit and read the auxiliary state.

- Base updates do not depend on R: the stages have an acyclic dependency.
- Triton kernels build on FLA and return intermediate residual information.
- Fixed dimensions: $\mathcal{O}(T)$ sequence work; fixed-size decoding state.
- At the same head shape, two matrices replace one: state capacity increases.

Paper, Sec. 4.1, PDF p. 5. Stage ordering explains the dependency; state-count comparison follows the equations.

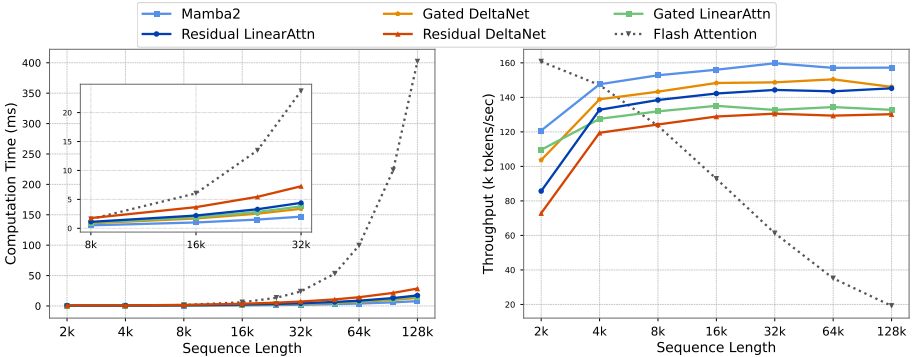
Main results use roughly 1.5B parameters and 100B tokens

Property	Transformer	Linear attention
Parameters	1.51B	1.55B
Hidden / intermediate	2,048 / 8,192	2,048 / 8,192
Heads / GQA groups	32 / 4	16 / 16
Head dimension	128	128
Layers	16 softmax	16 linear

- Baselines: sGLA for RLA, GDN for RDN; RetNet/Mamba2; Transformer reference.
- Shared AdamW and cosine schedule, 4M-token batch, $c = 1$.
- Training corpus and context length are not specified in this version.

Paper, Sec. 4.1, PDF pp. 5–6; App. C, Tables 6–7, PDF p. 15.

Kernel scaling remains linear from 2K to 128K



- Time (left, lower is better); throughput (right, higher is better).
- Extra fitting costs computation; long-sequence throughput stays relatively flat.
- Hardware/batch/precision details are absent; 128K is an efficiency test length.

Original Fig. 2, PDF p. 6. Do not infer 128K retrieval accuracy or exact speedups from this plot.

RDN leads the language-modeling and task averages

Model	Wiki ppl ↓	LAMB ppl ↓	Eight-task acc. (%) ↑
Transformer	17.33	19.53	46.51
RetNet	18.86	27.62	43.69
Mamba2	18.42	20.80	45.54
sGLA	17.63	18.06	45.79
GDN	17.27	15.76	46.03
RLA	17.35	15.59	46.44
RDN	**16.57**	**14.93**	**47.20**

- RLA vs. sGLA: +0.65 pp mean accuracy; RDN vs. GDN: +1.17 pp.
- Mean mixes raw accuracy and normalized accuracy, as specified by task.

Paper, Table 2, PDF p. 7; 100B-token main experiments.

Aggregate gains include task-specific losses

Task accuracy (%)	GDN	RDN	Difference (pp)
ARC-Easy	54.0	58.7	+4.7
LAMBADA	44.0	48.7	+4.7
WinoGrande	51.9	49.5	−2.4
SocialIQA	38.1	37.7	−0.4

- RDN's overall mean exceeds the Transformer reference by 0.69 pp.
- Transformer still scores 53.5% on WinoGrande, versus RDN's 49.5%.
- Reported values are point estimates; independent training variance is not given.

Paper, Table 2, PDF p. 7. Differences computed from reported values.

RLA excels on NIAH; RDN has the stronger recall mean

Model	DROP	FDA	NIAH	Seven-task mean
Transformer	26.9	63.0	70.5	48.31
sGLA	25.7	51.8	76.6	45.06
GDN	25.9	47.4	75.7	44.99
RLA	26.5	51.5	**83.6**	46.73
RDN	**27.8**	57.5	79.2	47.11

- Accuracy (%); mean includes DROP/FDA/NQ/SQD/SWDE/TQA and NIAH.
- RLA gains 7.0 pp NIAH over sGLA; RDN gains 10.1 pp FDA over GDN.
- RDN's overall mean still trails Transformer; detailed NIAH protocol is absent.

Paper, Table 3, PDF p. 7; task description in Sec. 4.2, PDF p. 6. Selected columns.

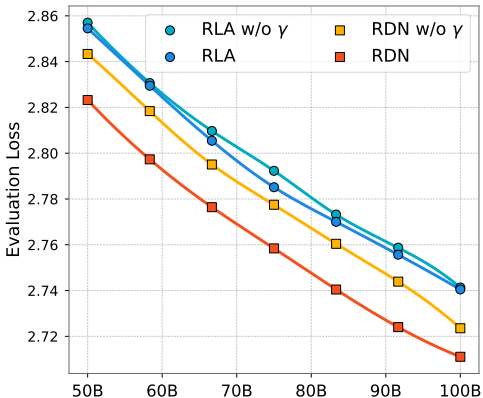
Persistent errors help most perplexities, with an exception

50B variant	Train loss	Wiki ppl	LAMB ppl	GSM8k ppl	HumanEval ppl
RLA	**2.22**	**18.76**	23.44	**3.92**	**9.61**
No fitting	2.26	20.19	**22.50**	6.85	16.23

- No fitting: replace persistent R_t with $(v_t - S_{t-1}k_t)k_t^\top$.
- All metrics are lower-is-better; LAMBADA favors the stateless variant.
- GSM8k/HumanEval measure perplexity here, not solve rate or pass@1.
- The table supports a broad benefit, narrower than the paragraph's wording.

Paper, Table 4 and Sec. 4.3, PDF p. 7. 50B-token ablation, distinct from main results.

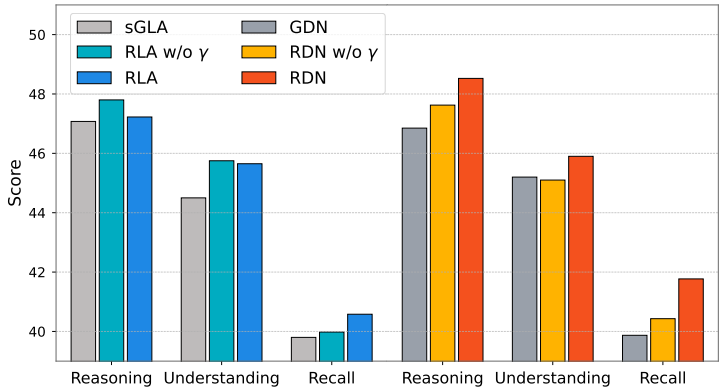
An independent correction rate lowers validation loss



- x: training tokens, 50B–100B.
- y: WikiText evaluation loss; lower is better.
- "w/o γ" means tying γ to β.
- Independent γ improves both variants, more visibly for RDN.

Original Fig. 3a and Sec. 4.3, PDF p. 8. Curve values are not digitized into exact claims.

The dedicated gate's task gains vary by category



- Higher is better; groups are Reasoning, Understanding, and Recall.
- RDN benefits across these groups; RLA Understanding shows no positive gain.

Original Fig. 3b, PDF p. 8. Bars compare baseline, tied correction rate, and independent correction rate.

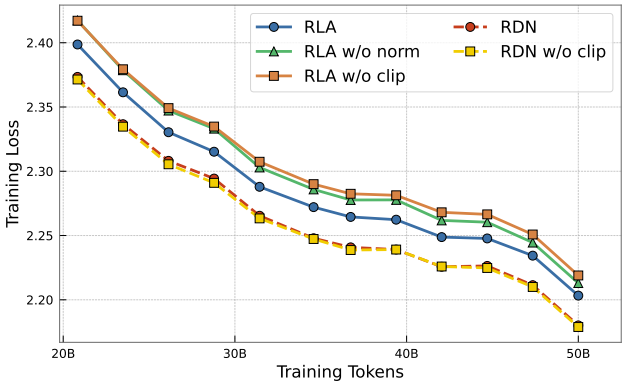
Output gates are less essential than residual memory

50B model	Output combination	Mean accuracy (%)
RLA	$\alpha_t S_{t-1} q_t + \gamma_t R_t q_t$	43.30
RLA	$S_{t-1} q_t + R_t q_t$	**43.81**
RDN	$\alpha_t S_{t-1} q_t + \gamma_t R_t q_t$	**44.65**
RDN	$S_{t-1} q_t + R_t q_t$	43.96

- Removing output gates improves RLA by 0.51 pp, reduces RDN by 0.69 pp.
- Both variants retain auxiliary-state learning in this ablation.
- This intervention differs from tying $\gamma = \beta$ in state learning.

Paper, Table 5, PDF p. 8. Same 50B-token budget and hyperparameters within this ablation.

Clipping and normalization matter more for additive RLA



- RLA without normalization or clipping has higher training loss.
- RDN and RDN without clipping nearly overlap.
- Delta updates empirically reduce sensitivity to residual clipping.

Original Fig. 4b, PDF p. 9. Training loss across roughly 20B–50B tokens; activation evidence in backup.

The practical comparison needs state and compute budgets

- **Capacity:** two same-shaped matrices replace one; no equal-state-budget control is reported.
- **Cost:** extra fitting preserves linear scaling but adds constant overhead.
- **Protocol:** corpus, training length, detailed retrieval settings, and full kernel configuration are unspecified in v1.
- **Attribution:** compare a wider single state against two residual-fitting states at matched memory/compute.

Reading assessment

A modular memory-learning mechanism with useful gains; the best deployment choice depends on task and budget.

Analysis from Secs. 3–4 and App. C–D, PDF pp. 4–9, 14–15.

Remembering errors is a useful axis for memory design

1. **Mechanism:** a persistent key-to-residual map complements the base key-to-value map.
2. **Computation:** additive or delta inner updates fit robust pseudo-residuals with reusable linear-attention kernels.
3. **Evidence:** RDN leads aggregate language metrics; RLA leads NIAH; single-task and gating ablations reveal tradeoffs.

Paper, Secs. 3–6, PDF pp. 3–9. Main talk: approximately 25 minutes; backup details follow.

References and version identity

- Lai et al. *Enhancing Linear Attention with Residual Learning*.
  arXiv:2509.25223v1, 24 September 2025.
  All equations, figures, tables, and PDF locators use this version.
- EMNLP 2026 main proceedings record.
  DOI: 10.18653/2026.emnlp-main.2394. Publication checked 13 November 2026.
- No paper-specific public implementation found on this lookup.
  Appendix D provides recurrent RLA pseudocode.

Figures use original author TeX assets. Author spelling follows arXiv metadata; v1 PDF spells Kang's given name "Jiangliang".

Backup: outer targets and inner updates can be separated

$$r_t = -\frac{\partial \mathcal{L}_{\text{outer}}(f(k_t; S_{t-1}), v_t)}{\partial f(k_t; S_{t-1})}$$

$$\begin{aligned} R_t &= R_{t-1} - \nabla_R \mathcal{L}_{\text{inner}}(f(k_t; R_{t-1}), r_t), \\ S_t &= S_{t-1} - \nabla_S \mathcal{L}_{\text{inner}}(f(k_t; S_{t-1}), v_t), \\ o_t &= f(q_t; S_{t-1}) + f(q_t; R_t). \end{aligned}$$

- Gating omitted here, as in the general framework.
- Outer mapping loss chooses the error signal; inner loss chooses additive/delta fitting.
- This local outer loss is distinct from end-to-end language-model training loss.

Paper, App. A, PDF p. 13. Online gradient-boosting interpretation.

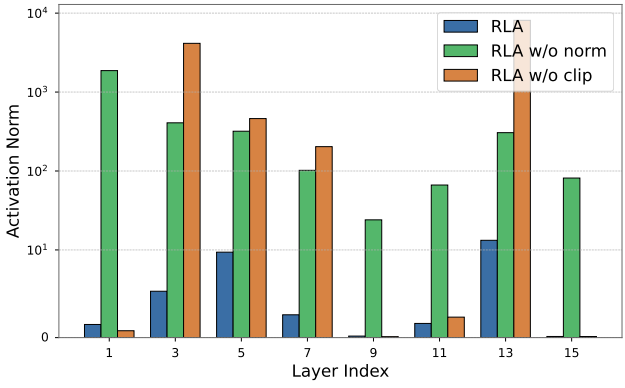
Backup: update order and the appendix sign inconsistency

1. Normalize q_t, k_t and read the old base state to form r_t.
2. Update R_t, then combine old S_{t-1} with updated R_t.
3. Update S_t for the next token.

- Pseudocode uses row vectors and transposed states: $k_t^\top v_t$, $q_t S$.
- Appendix B prints two *subtractive* outer-product writes.
- For $\mathcal{L}_{\text{inner}} = -\langle Rk, r \rangle$, gradient descent gives $R_t = R_{t-1} + r_t k_t^\top$.
- Main formulas and pseudocode use the consistent additive sign.

Paper, Sec. 3.4, PDF p. 5; App. B, PDF p. 14; App. D, Listing 1, PDF p. 15.

Backup: missing safeguards enlarge RLA activations



- Layer index (x); attention-output L2 norm (y).
- Without normalization/clipping, several layers reach norms of 10^2–10^3.

Original Fig. 4a, PDF p. 9. Complements the training-loss curves in the main talk.

Backup: training hyperparameters and state accounting

Property	Value
Peak / minimum learning rate	3×10^{-4} / 3×10^{-5}
Batch / warmup / total tokens	4M / 0.5B / 100B
Weight decay / gradient clipping	0.1 / 1.0
Initialization standard deviation	0.006

- 16 heads, 128-dimensional keys/values: 262,144 elements per state per layer.
- Two states: 524,288 elements per layer, derived from the reported shape.
- These element counts do not assume a storage dtype or measure peak memory.

Paper, App. C, Tables 6–7, PDF p. 15. State counts calculated from model dimensions.