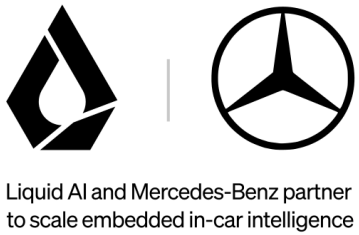
On Linear Attention

Theoretical Perspectives on Architecture Design

Philipp Nazari

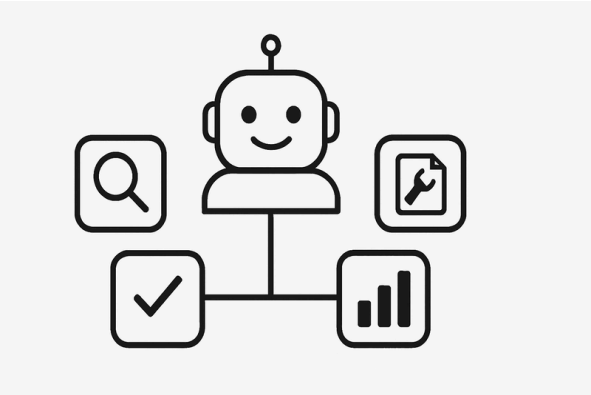
June 25, 2026

Deployment in resource-constrained **environments**



Liquid AI and Mercedes-Benz partner to scale embedded in-car intelligence

Long context scenarios



Softmax Attention as a Sequence Model

Receive a sequence $x_1, x_2, ..., x_L$ of length L and map it to another sequence:

$$(\boldsymbol{x}_1, ..., \boldsymbol{x}_L) \to (\boldsymbol{o}_1, ..., \boldsymbol{o}_L).$$

A particular instance is **softmax attention**.

Input	→	**Project**	→	**Attention**	→	**Output**
$\boldsymbol{X}$		$\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V} = \boldsymbol{X}\boldsymbol{W}_{\boldsymbol{Q},\boldsymbol{K},\boldsymbol{V}}$		$\boldsymbol{A} = \mathrm{softmax}(\boldsymbol{Q}\boldsymbol{K}^\top \odot \boldsymbol{M})$		$\boldsymbol{O} = \boldsymbol{A}\boldsymbol{V}$

Softmax Attention: Autoregressive Generation

During training, use the parallel realization of the softmax operator:

$$\boldsymbol{O} = \mathrm{softmax}(\boldsymbol{Q}\boldsymbol{K}^\top \odot \boldsymbol{M})\boldsymbol{V}$$

During autoregressive decode, need to use the recurrent formulation:

$$\boldsymbol{o}_t = \sum_{i=1}^{t-1} \left(\frac{\exp(\boldsymbol{q}_t^\top \boldsymbol{k}_i)}{\sum_{j=1}^{t} \exp(\boldsymbol{q}_t^\top \boldsymbol{k}_j)} \right) \boldsymbol{v}_i + \left(\frac{\exp(\boldsymbol{q}_t^\top \boldsymbol{k}_t)}{\sum_{j=1}^{t} \exp(\boldsymbol{q}_t^\top \boldsymbol{k}_j)} \right) \boldsymbol{v}_t.$$

Softmax Attention: Autoregressive Generation

During training, use the parallel realization of the softmax operator:

$$\boldsymbol{O} = \mathrm{softmax}(\boldsymbol{Q}\boldsymbol{K}^\top)\boldsymbol{V}$$

During autoregressive decode, need to use the recurrent formulation:

$$\boldsymbol{o}_t = \sum_{i=1}^{t-1} \left(\frac{\exp(\boldsymbol{q}_t^\top \boldsymbol{k}_i)}{\sum_{j=1}^{t-1} \exp(\boldsymbol{q}_t^\top \boldsymbol{k}_j) + \exp(\boldsymbol{q}_t^\top \boldsymbol{k}_t)} \right) \boldsymbol{v}_i + \left(\frac{\exp(\boldsymbol{q}_t^\top \boldsymbol{k}_t)}{\sum_{j=1}^{t-1} \exp(\boldsymbol{q}_t^\top \boldsymbol{k}_j) + \exp(\boldsymbol{q}_t^\top \boldsymbol{k}_t)} \right) \boldsymbol{v}_t.$$

In practice, one keeps a **KV-cache** to avoid re-computation: $\mathcal{O}(L)$ **memory**

Can we do better? Yes, with **linear Attention :** $\mathcal{O}(1)$ **memory**

Linear Attention: Parallel Formulation

Remove the softmax operator:[1]

$$\boldsymbol{O} = (\boldsymbol{Q}\boldsymbol{K}^\top)\boldsymbol{V}.$$

Softmax form.

$$\mathrm{softmax}(\boldsymbol{Q}\boldsymbol{K}^\top)\boldsymbol{V}$$

The nonlinear row-wise normalization blocks reassociation.

Linear form.

$$\boldsymbol{Q}(\boldsymbol{K}^\top\boldsymbol{V})$$

The history can be summarized before seeing the query.

[1] Katharopoulos et al., *Transformers are RNNs: Fast Autoregressive Transformers with Linear Attention*, ICML 2020.

Linear Attention: Recurrent Formulation

At timestep t,

$$\boldsymbol{o}_t = \sum_{i \le t} \boldsymbol{v}_i (\boldsymbol{k}_i^\top \boldsymbol{q}_t) = \left(\sum_{i \le t} \boldsymbol{v}_i \boldsymbol{k}_i^\top \right) \boldsymbol{q}_t.$$

Define the hidden state

$$\boldsymbol{S}_t := \sum_{i \le t} \boldsymbol{v}_i \boldsymbol{k}_i^\top \in \mathbb{R}^{d_v \times d_k}.$$

which yields the recursion

$$\boldsymbol{S}_t = \boldsymbol{S}_{t-1} + \boldsymbol{v}_t \boldsymbol{k}_t^\top, \quad \boldsymbol{o}_t = \boldsymbol{S}_t \boldsymbol{q}_t.$$

Linear Attention is a **linear RNN!**

Linear Attention: Problem

Linear attention writes

$$\boldsymbol{S}_t = \boldsymbol{S}_{t-1} + \boldsymbol{v}_t \boldsymbol{k}_t^\top .$$

What can go wrong?

Memory conflict! Two timesteps writing into the same memory

If a later token has a similar key, the model writes another value into nearly the same memory direction.

Read the current memory at key $\boldsymbol{k}_t$:

$$\hat{\boldsymbol{v}}_t = \boldsymbol{S}_{t-1}\boldsymbol{k}_t.$$

Plain linear attention

$$\boldsymbol{S}_t = \boldsymbol{S}_{t-1} + \boldsymbol{v}_t\boldsymbol{k}_t^\top$$

The prediction error is

$$\boldsymbol{e}_t = \boldsymbol{v}_t - \boldsymbol{S}_{t-1}\boldsymbol{k}_t.$$

DeltaNet writes this residual:

$$\boldsymbol{S}_t = \boldsymbol{S}_{t-1} + \boldsymbol{e}_t\boldsymbol{k}_t^\top = \boldsymbol{S}_{t-1}\left(\boldsymbol{I} - \boldsymbol{k}_t\boldsymbol{k}_t^\top\right) + \boldsymbol{v}_t\boldsymbol{k}_t^\top.$$

Different memory dynamics

Model	Memory update	Adds
Linear Attention[1]	$\boldsymbol{S}_t = \boldsymbol{S}_{t-1} + \boldsymbol{v}_t \boldsymbol{k}_t^\top$	write
Mamba-2[3]	$\boldsymbol{S}_t = \alpha_t \boldsymbol{S}_{t-1} + \boldsymbol{v}_t \boldsymbol{k}_t^\top$	forget
DeltaNet[2]	$\boldsymbol{S}_t = \boldsymbol{S}_{t-1}(\boldsymbol{I} - \boldsymbol{k}_t \boldsymbol{k}_t^\top) + \boldsymbol{v}_t \boldsymbol{k}_t^\top$	erase
Gated DeltaNet[4]	$\boldsymbol{S}_t = \alpha_t \boldsymbol{S}_{t-1}(\boldsymbol{I} - \boldsymbol{k}_t \boldsymbol{k}_t^\top) + \boldsymbol{v}_t \boldsymbol{k}_t^\top$	forget + erase

[1] Katharopoulos et al., ICML 2020.
[3] Dao & Gu, *Transformers are SSMs*, 2024.
[2] Schlag et al., ICML 2021.
[4] Yang, Kautz, Hatamizadeh, *Gated Delta Networks*, 2024/2025.

Our Research: Efficient Efficient Architectures

Linear Recurrent Unit[5]:

$$\boldsymbol{h}_t = \boldsymbol{A}\boldsymbol{h}_{t-1} + \boldsymbol{B}\boldsymbol{x}_t \in \mathbb{R}^d, \qquad \boldsymbol{y}_t = \boldsymbol{C}\boldsymbol{h}_t$$

$\boldsymbol{h}_t$ acts as a fixed-size **summary of the past**

- **Questions**:
  1. How well does model **utilize** this fixed-size hidden state?
  2. Can we make it more **efficient**?
- We want **large states** for better memory, but **small states** for efficiency.

[5] A. Orvieto et al. Resurrecting Recurrent Neural Networks for Long Sequences.

Project 1: Controllability & Observability Gramians

- **identify** and **truncate** unused state dimensions during training

[6]

- Solve the discrete Lyapunov equations for the **Gramians**:

$$APA^\top + BB^\top = P, \qquad A^\top QA + C^\top C = Q$$

- $P =$ **Controllability** (input - hidden state)
- $Q =$ **Observability** (hidden state - output)

A state dimension that is hard to reach **and** hard to observe is essentially dead weight.

---

[6] Makram Chahine, **Philipp Nazari**, Daniela Rus, T. Konstantin Rusch. *CompreSSM: The Curious Case of In-Training Compression of State Space Models*. arXiv preprint arXiv:2510.02823 (2025).

- **Hankel Singular Values (HSVs)** combine both perspectives into a single importance ranking:

$$\sigma_i = \sqrt{\lambda_i(\boldsymbol{PQ})}, \qquad \sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_n$$

- **Balanced realization**: transform coordinates. States with small σ_i are both hard to control and hard to observe.
- **Truncation**: discard the tail of the balanced system.

Project 1: The CompreSSM Algorithm

- **The Pipeline**. During the **first 10%** of training:
  1. Extract matrices and compute HSVs periodically.
  2. Sort and truncate the state dimension r based on an energy threshold.
  3. Replace weights and continue training.

Project 1: CompreSSM Results (CIFAR10)

- **performance close to a full-size model** at the **speed and memory footprint** of a much smaller one.

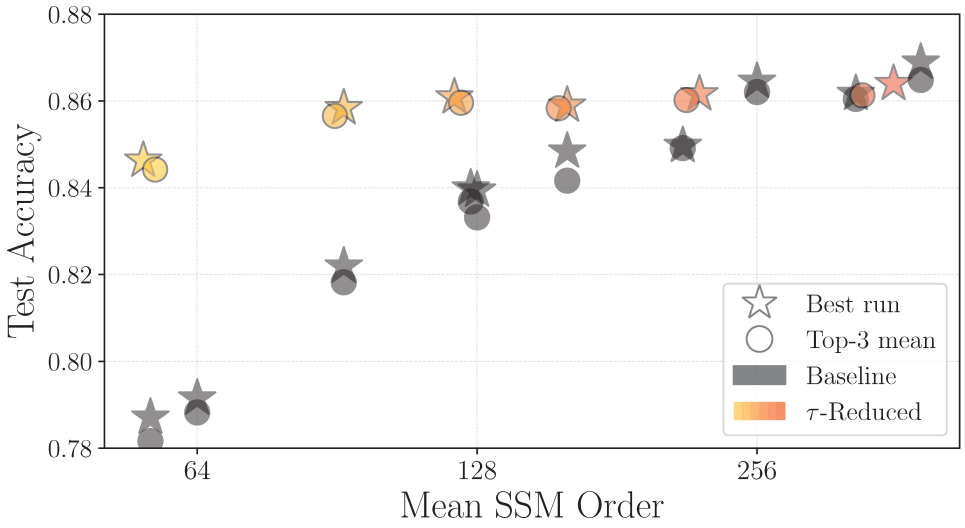


Figure 1: Test Acc vs. Final Dim

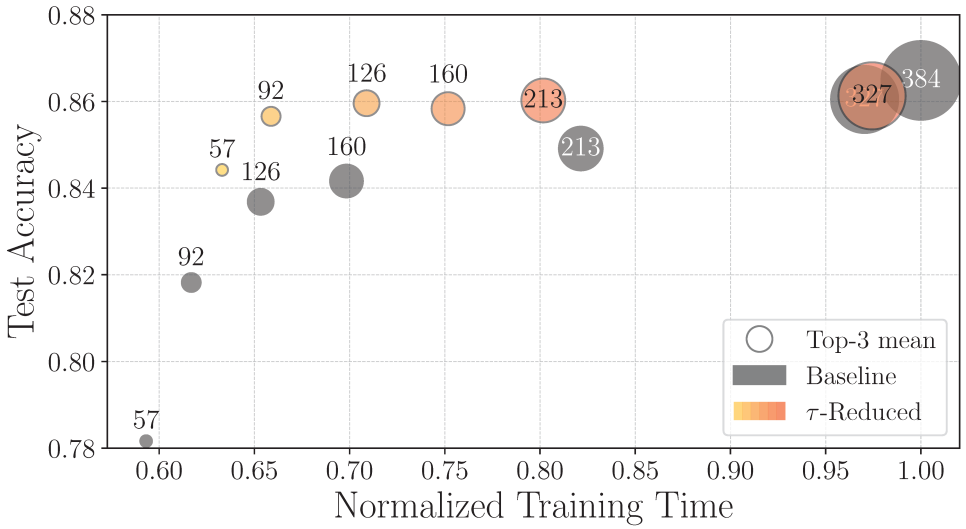
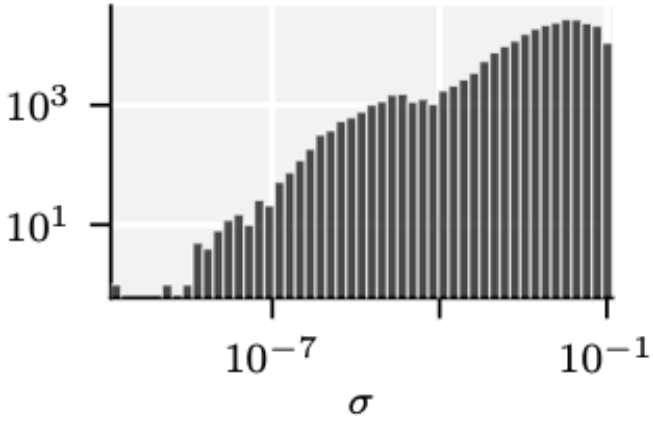


Figure 2: Test Acc vs. Train Time

In trained models, spectrum of S_t is often heavy-tailed.[7]

For plain linear attention,

$$\boldsymbol{S}_t = \sum_{i \le t} \boldsymbol{v}_i \boldsymbol{k}_i^\top.$$



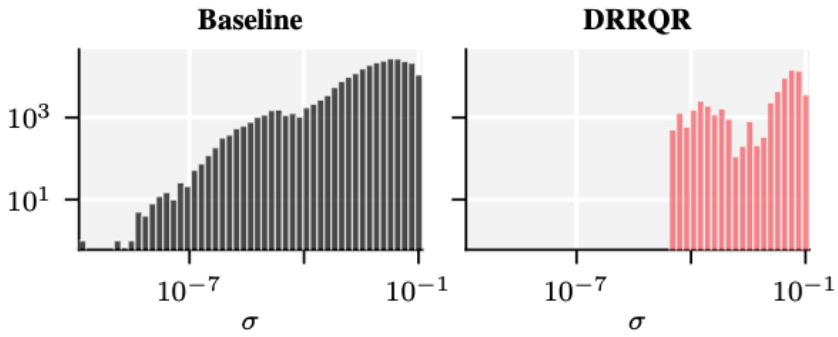
Therefore

$$\operatorname{rank}(\boldsymbol{S}_t) \le \min(\operatorname{rank}(\boldsymbol{V}_t), \operatorname{rank}(\boldsymbol{K}_t)).$$

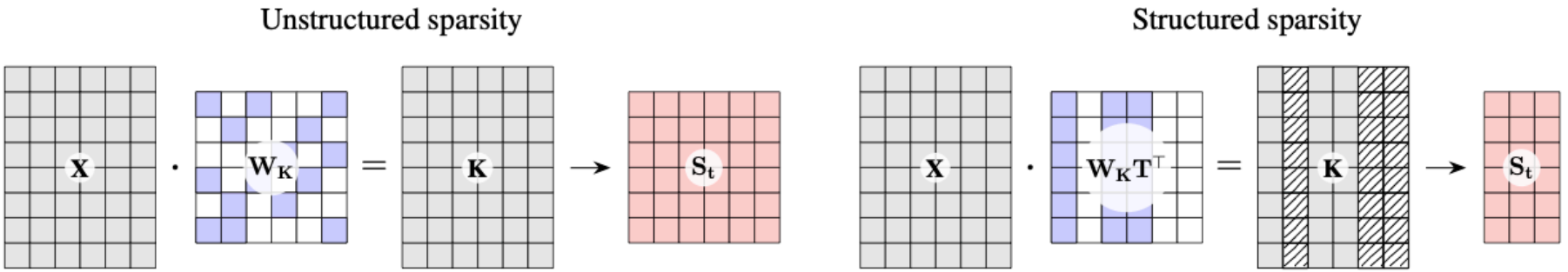
[7] **Nazari** & Rusch, *The Key to State Reduction in Linear Attention: A Rank-Based Perspective*, 2026.

Project 2: Rank-based state reduction

Prune query/key dimension in a rank-aware manner: choose a subset of key/query channels shrinking the state to



$$\boldsymbol{S}_t \in \mathbb{R}^{d_v \times d'_k} \text{ with } d_{k'} < d_k$$



Project 2: Results

Result

Around **50% of key/query channels** can be removed with only marginal perplexity cost, while reducing state size and improving sequence-mixer throughput.

Table 1: Comprehensive post-RFT evaluation of Gated DeltaNet 1.3B. We report Wikitext and Lambada perplexities, the average zero-shot generation accuracy on language-model evaluations (**LM evals**, [16]) across varying compression ratios (best results in **bold**, second best underlined).

	75%			50%			40%			30%		
Method	Wiki ↓	LMB ↓	LM Evals ↑	Wiki ↓	LMB ↓	LM Evals ↑	Wiki ↓	LMB ↓	LM Evals ↑	Wiki ↓	LMB ↓	LM Evals ↑
Rand	19.3	18.1	54.6	16.9	11.5	57.8	16.5	10.7	58.3	16.1	10.2	58.7
L1	18.6	16.8	55.3	16.8	13.0	57.2	16.5	11.9	57.8	16.3	11.3	58.3
S-Wanda	18.2	15.1	55.4	16.6	11.7	57.8	16.4	11.5	58.1	16.1	11.0	58.6
Grad	**17.8**	**12.5**	**56.8**	16.4	**10.5**	**58.3**	**16.1**	**10.1**	58.6	**15.9**	**9.9**	**59.0**
DRRQR	17.9	14.7	56.2	**16.3**	12.0	58.0	**16.1**	11.0	**58.7**	**15.9**	10.7	58.8
Baseline	16.8	9.7	59.4	16.8	9.7	59.4	16.8	9.7	59.4	16.8	9.7	59.4

Thanks! Questions?

References

- Vaswani et al., *Attention Is All You Need*, NeurIPS 2017.
- Katharopoulos et al., *Transformers are RNNs: Fast Autoregressive Transformers with Linear Attention*, ICML 2020.
- Schlag, Irie, Schmidhuber, *Linear Transformers Are Secretly Fast Weight Programmers*, ICML 2021.
- Dao & Gu, *Transformers are SSMs: Generalized Models and Efficient Algorithms through Structured State Space Duality*, 2024.
- Yang, Kautz, Hatamizadeh, *Gated Delta Networks: Improving Mamba2 with Delta Rule*, 2024/2025.
- Chahine, Nazari, Rus, Rusch, *The Curious Case of In-Training Compression of State Space Models*, ICLR 2026.
- Nazari & Rusch, *The Key to State Reduction in Linear Attention: A Rank-Based Perspective*, 2026.