FUNCTION-VALUED MEMORY: QUERY-DEPENDENT VALUES FOR LINEAR ATTENTION

Anonymous authors
Paper under double-blind review

ABSTRACT

Matrix-state linear attention compresses history into fixed-size recurrent memory. Its design involves both update dynamics and the Query-to-Value function class represented by the fast state; under a matrix-scale budget, expanding the latter becomes a capacity-allocation problem. We introduce **Function-Valued Linear Attention (FVLA)**, which generalizes the *fixed Value vector* associated with each memory coordinate to a *Query-dependent response function*: the Query determines both how strongly the coordinate is read and what Value it returns. A separable quadratic parameterization learns response directions across contexts while adapting their Value coefficients online within each context. Although nonlinear in the Query, FVLA remains linear in its online coefficients, preserving shared-error Delta updates and chunkwise computation. Across Gated DeltaNet and Kimi Delta Attention experiments spanning 140M and 340M configurations, FVLA improves language modeling and in-context retrieval. Across the four host–scale pairs, one and four response modes yield median relative gains of 10.5% and 24.0%, respectively, in average accuracy over the six retrieval tasks. Ablations show that nonlinear responses use additional fast-state capacity more effectively than linear state expansion. Response-mode scaling yields a clear *quality–state–compute trade-off*, with overall modeling quality improving as more capacity is allocated to the response space. Models trained at 2K context retain language-modeling gains through 512K-token streaming evaluation and improve long-range retrieval beyond training length. Together, these results identify response-space allocation as a key design dimension for finite-state associative memory.

1 INTRODUCTION

Self-attention performs context-dependent retrieval through interactions between queries and historical key–value pairs, but standard Softmax Attention requires $\mathcal{O}(L^2)$ computation for training and prefill and an $\mathcal{O}(L)$ KV cache during autoregressive decoding (Vaswani et al., 2017). Linear Attention instead compresses the growing history into a fixed-size recurrent state, allowing sequence computation to scale linearly with context length (Katharopoulos et al., 2020; Cerruti et al., 2026). For matrix-state variants, this recurrent state can be viewed as continually learning key–value associations within a finite memory. Recent work has largely focused on improving how those associations are updated, for example through prediction-error correction in DeltaNet (Yang et al., 2024c) and input-dependent retention in Gated DeltaNet (Yang et al., 2024a). Viewed through Test-Time Training(TTT) and Test-Time Regression(TTR), the state itself acts as a fast learner that fits key-to-value associations online (Sun et al., 2025; Wang et al., 2026). This perspective motivates separating two design questions for finite matrix memories: *update dynamics*, which determine how fast weights evolve with new observations, and the *response space*, which determines what Query-to-Value functions a fixed fast state can represent. Many recent matrix-state fast learners improve the former while retaining a linear memory readout, $f_t(\boldsymbol{q}) = \boldsymbol{S}_t\boldsymbol{q}$. Better optimization, or more state allocated to the same linear readout, can therefore increase fast-state capacity without enlarging the function class of the fixed-state memory response.

A natural way to enlarge this response space is to use nonlinear fast learners or higher-order query features, as explored by methods such as TTT-MLP (Sun et al., 2025) and ATLAS (Behrouz et al., 2026b). A complete second-order response space, however, is expensive: representing all pairwise interactions of a d-dimensional Query requires $\mathcal{O}(d^2)$ features and $\mathcal{O}(d^2 d_v)$ fast-state capacity when

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

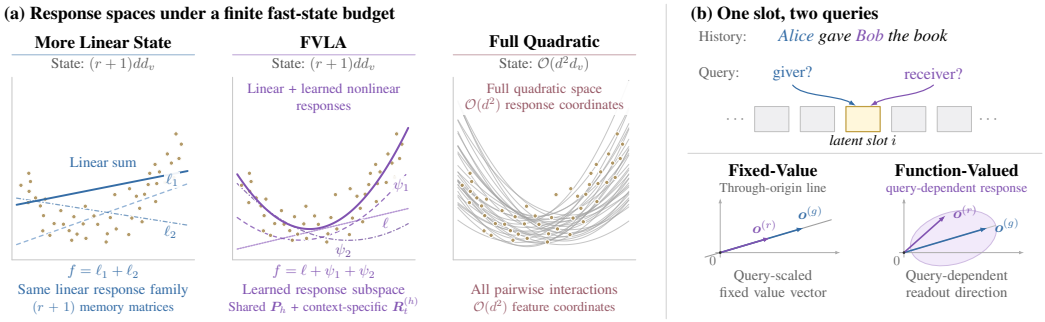


Figure 1: **Response-space allocation under a limited fast-state budget.** a) Additional linear states increase storage capacity but preserve a linear fixed-state response, while FVLA allocates limited capacity to structured nonlinear responses; representing the complete second-order space instead requires $\mathcal{O}(d^2 d_v)$ fast-state coefficients. b) For a fixed state, a standard matrix-memory coordinate returns a fixed Value vector scaled by the Query, whereas FVLA allows the returned Value itself to vary with the Query through shared response directions and context-specific fast coefficients.

predicting a d_v-dimensional Value. Under a limited fast-state budget, this raises a different question from simply enlarging memory capacity: *should additional state be spent on more linear responses, or on a small number of more useful nonlinear responses?* FVLA takes this approach by using the additional state to represent a small set of structured Query-dependent responses, allowing the compressed memory to adapt how stored information is translated into Values for different Queries.

We therefore introduce *Function-Valued Linear Attention* (FVLA), where each memory coordinate produces a Value that varies as a function of the current Query, as illustrates in Figure 1. FVLA parameterizes this Query dependence using r quadratic response modes. For the h-th mode, FVLA uses the quadratic feature $\psi_{P_h}(\boldsymbol{q}) = c_h[\boldsymbol{q} \odot (\boldsymbol{P}_h \boldsymbol{q})]$, where $\boldsymbol{P}_h$ is a learned linear map, c_h controls the feature scale, and $\odot$ denotes elementwise multiplication. Its i-th component is $c_h q_i (\boldsymbol{p}_{h,i}^\top \boldsymbol{q})$, allowing the Value response of memory coordinate i to depend on a learned combination of all Query dimensions rather than only on q_i. Each mode is paired with a matrix state that is updated online to map these Query interactions to Values in the current context. Additional modes enlarge the family of Query-dependent responses the memory can represent, while increasing state size and computation. Since these matrix states enter the readout linearly, they can be updated together with the original linear memory using the same Delta-style prediction error.

Our contributions are:

- We identify a representational bottleneck in matrix-state linear attention, where a fixed memory is restricted to linear Query-to-Value responses regardless of how the state is updated.
- We introduce FVLA, which turns each fixed-value memory coordinate into a Query-dependent value function, expanding the memory's response space while retaining matrix-scale fast states and Delta-style online updates.
- We show across GDN and KDA that allocating recurrent state to these nonlinear responses improves language modeling and retrieval, and is more effective under matched state budgets than expanding linear memory alone.

2 Preliminaries And Motivation: Response Spaces of Test-Time Learners

Attention as a Test-Time Learner Test-Time Regression (TTR) views Attention and recurrent associative memories as regressors formed from the current context, with different mechanisms characterized by how historical associations are weighted, which function class the regressor represents, and how it is optimized at test time (Wang et al., 2026). Test-Time Training (TTT) provides a parameterized realization of this view, where the recurrent state itself is a fast learner whose parameters are updated online (Sun et al., 2025). Let $f(\cdot; \boldsymbol{W}_t)$ denote such a learner, with Key $\boldsymbol{k}_t$, Value

$\boldsymbol{v}_t$, and Query $\boldsymbol{q}_t$ obtained from learned projections of the current token representation. Its online learning and readout can be written as

$$\boldsymbol{W}_t = \mathcal{U}(\boldsymbol{W}_{t-1}; \boldsymbol{k}_t, \boldsymbol{v}_t), \qquad \boldsymbol{o}_t = f(\boldsymbol{q}_t; \boldsymbol{W}_t). \tag{1}$$

We call the family of Query-to-Value functions realizable as the fast state varies the learner's *response space*. For a matrix-valued fast learner, $f(\boldsymbol{q}; \boldsymbol{S}) = \boldsymbol{S}\boldsymbol{q}$, this space is $\mathcal{F}_{\text{lin}} = \{\boldsymbol{q} \mapsto \boldsymbol{S}\boldsymbol{q} : \boldsymbol{S} \in \mathbb{R}^{d_v \times d}\}$. Thus, once the history and hence $\boldsymbol{S}_t$ are fixed, a matrix-state Linear Attention memory can only produce a linear response to the Query, regardless of how sophisticated its online state update is (Schlag et al., 2021; Sun et al., 2025). This separates two design dimensions of associative memory: how the fast state is updated, and what Query-to-Value responses that state can represent.

Query-dependent messages in aggregated history. Softmax Attention provides a useful contrast because its returned content is itself Query-dependent. Writing it as Nadaraya–Watson regression with $\kappa(\boldsymbol{q}, \boldsymbol{k}) = \exp(\boldsymbol{q}^\top \boldsymbol{k}/\sqrt{d})$ (Nadaraya, 1964; Watson, 1964), consider any partition of the history into subsets $\{\mathcal{C}_i\}_{i=1}^M$ and define

$$Z_i(\boldsymbol{q}) = \sum_{j \in \mathcal{C}_i} \kappa(\boldsymbol{q}, \boldsymbol{k}_j), \qquad \boldsymbol{m}_i(\boldsymbol{q}) = \frac{\sum_{j \in \mathcal{C}_i} \kappa(\boldsymbol{q}, \boldsymbol{k}_j)\boldsymbol{v}_j}{Z_i(\boldsymbol{q})}. \tag{2}$$

The attention output can then be written exactly as

$$\boldsymbol{o}_t = \sum_{i=1}^{M} \frac{Z_i(\boldsymbol{q}_t)}{\sum_{\ell=1}^{M} Z_\ell(\boldsymbol{q}_t)} \boldsymbol{m}_i(\boldsymbol{q}_t). \tag{3}$$

This decomposition exposes two roles of the Query: $Z_i(\boldsymbol{q})$ determines *how much* an aggregated unit contributes, while $\boldsymbol{m}_i(\boldsymbol{q})$ determines *what* content it returns. When multiple Key–Value associations are aggregated, their relative weights vary with the Query, so the returned message is generally a function of the Query itself. Standard matrix memory retains only the first degree of freedom: each coordinate stores a fixed Value vector whose contribution is scaled by the Query. This motivates assigning each memory coordinate a Query-dependent response, which FVLA realizes through learned nonlinear features.

3 Function-Valued Linear Attention

FVLA augments matrix-state Linear Attention with structured Query-dependent responses while retaining matrix-valued fast states. Figure 2 gives an overview of the construction and its Gated DeltaNet instantiation. The following subsections derive the response parameterization, show how its linear and nonlinear states are learned jointly online, and characterize the resulting state–compute trade-off and chunkwise computation.

3.1 From Fixed-Value to Function-Valued Memory

Let $\boldsymbol{S}_t \in \mathbb{R}^{d_v \times d}$ denote a standard matrix memory. Its readout can be written as

$$f_t(\boldsymbol{q}) = \boldsymbol{S}_t \boldsymbol{q} = \sum_{i=1}^{d} q_i \, \boldsymbol{s}_{t,i}, \tag{4}$$

where $\boldsymbol{s}_{t,i} \in \mathbb{R}^{d_v}$ is the Value vector associated with the i-th response coordinate. For a fixed fast state, q_i controls only how strongly this coordinate is read; the returned content $\boldsymbol{s}_{t,i}$ itself does not depend on the Query. Standard matrix memory can therefore be viewed as a collection of fixed-value responses.

FVLA relaxes this restriction by allowing each response coordinate to return a function of the Query. We consider a first-order Query-dependent response at each original coordinate,

$$\boldsymbol{m}_{t,i}(\boldsymbol{q}) = \boldsymbol{s}_{t,i} + \boldsymbol{C}_{t,i}\boldsymbol{q}, \qquad \boldsymbol{C}_{t,i} \in \mathbb{R}^{d_v \times d},$$

so that

$$f_t(\boldsymbol{q}) = \sum_{i=1}^{d} q_i \, \boldsymbol{m}_{t,i}(\boldsymbol{q}) = \boldsymbol{S}_t \boldsymbol{q} + \sum_{i=1}^{d} q_i \, \boldsymbol{C}_{t,i}\boldsymbol{q}. \tag{5}$$

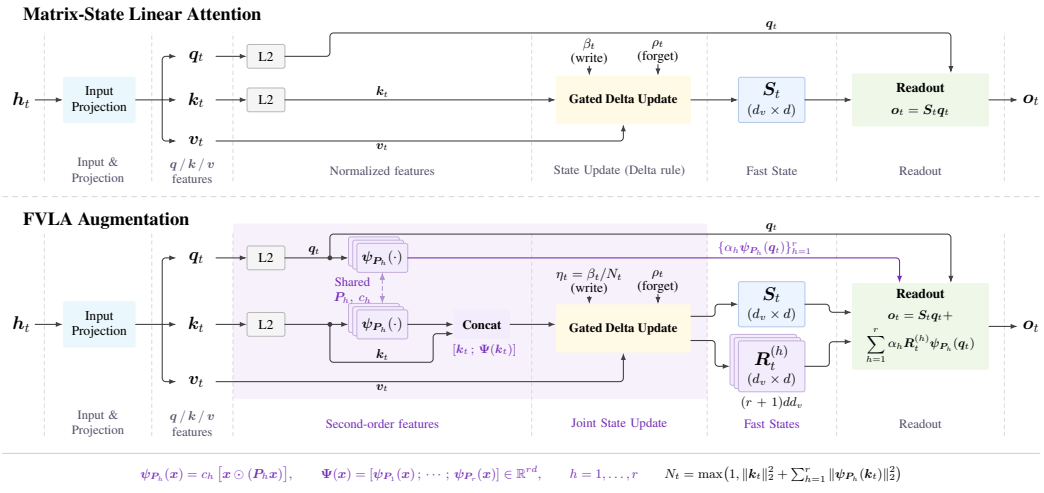


Figure 2: **Overview of FVLA and an instantiation on Gated DeltaNet.** Standard matrix-state Linear Attention stores a fast state $\boldsymbol{S}_t$ and reads out a linear response $\boldsymbol{S}_t\boldsymbol{q}_t$. FVLA adds Query-dependent response features $\boldsymbol{\psi}_{\boldsymbol{P}_h}(\cdot)$, whose structure is learned by slow parameters and whose Value coefficients are stored in fast states $\boldsymbol{R}_t^{(h)}$. The figure illustrates this construction and its instantiation on Gated DeltaNet: linear and nonlinear coefficient blocks are updated from the same Key-to-Value prediction error and jointly contribute to the final readout. FVLA changes the response parameterization while leaving the host state dynamics otherwise intact.

Here, q_i still determines *how much* is read from the i-th coordinate, while $\boldsymbol{m}_{t,i}(\boldsymbol{q})$ determines *what* that coordinate returns for the current Query. The quadratic Query interaction therefore arises naturally from replacing a fixed returned Value with a first-order Query-dependent response.

Maintaining all $\boldsymbol{C}_{t,i}$ online would require $\mathcal{O}(d^2 d_v)$ fast-state coefficients. FVLA keeps the additional state at matrix scale by representing these Query-dependent variations with r separable *response modes*. For mode h, we define

$$\begin{aligned} \boldsymbol{C}_{t,i}^{(h)} &= c_h\, \boldsymbol{r}_{t,i}^{(h)} \boldsymbol{p}_i^{(h)\top}, \\ \boldsymbol{C}_{t,i} &= \sum_{h=1}^{r} \alpha_h \boldsymbol{C}_{t,i}^{(h)} = \sum_{h=1}^{r} \alpha_h c_h\, \boldsymbol{r}_{t,i}^{(h)} \boldsymbol{p}_i^{(h)\top}. \end{aligned} \tag{6}$$

Here, $\boldsymbol{p}_i^{(h)} \in \mathbb{R}^d$ is a Query-modulation direction shared across contexts, $\boldsymbol{r}_{t,i}^{(h)} \in \mathbb{R}^{d_v}$ is its Value coefficient updated online within the current context, and c_h is a learned slow feature scale. The readout gain α_h controls how strongly the h-th nonlinear response mode contributes to the final output.

Let $\boldsymbol{P}_h \in \mathbb{R}^{d\times d}$ collect $\boldsymbol{p}_i^{(h)\top}$ as its rows, and let $\boldsymbol{R}_t^{(h)} \in \mathbb{R}^{d_v \times d}$ collect $\boldsymbol{r}_{t,i}^{(h)}$ as its columns. Defining the nonlinear response feature

$$\boldsymbol{\psi}_{\boldsymbol{P}_h}(\boldsymbol{q}) = c_h\big[\boldsymbol{q} \odot (\boldsymbol{P}_h \boldsymbol{q})\big] \in \mathbb{R}^d, \tag{7}$$

the FVLA readout becomes

$$\widetilde{f}_t(\boldsymbol{q}) = \boldsymbol{S}_t\boldsymbol{q} + \sum_{h=1}^{r} \alpha_h \boldsymbol{R}_t^{(h)} \boldsymbol{\psi}_{\boldsymbol{P}_h}(\boldsymbol{q}). \tag{8}$$

Equivalently, FVLA can be viewed as a linear fast learner over the original Query together with learned structured quadratic features.

Each response mode adds one $d_v \times d$ fast state, giving a total main fast-state size of $(r+1)dd_v$. Increasing r therefore expands the Query-dependent response space while state size and feature computation grow approximately linearly with the number of modes. The slow matrices $\boldsymbol{P}_h$ capture

response structure shared across contexts, whereas the fast states $\boldsymbol{R}_t^{(h)}$ adapt the corresponding Values online; c_h sets the feature scale and α_h calibrates the final readout. Because Equation 8 is nonlinear in the Query but linear in the fast states, all response components can be treated as a single expanded predictor and updated jointly, as developed next.

3.2 Joint Online Learning in the Expanded Response Space

Although FVLA is nonlinear in the Query, it remains linear in the fast states $\boldsymbol{S}_t$ and $\{\boldsymbol{R}_t^{(h)}\}_{h=1}^{r}$. We therefore collect the linear and nonlinear response features into an expanded write address and read address:

$$\boldsymbol{x}_t = \begin{bmatrix} \boldsymbol{k}_t \\ \boldsymbol{\psi}_{\boldsymbol{P}_1}(\boldsymbol{k}_t) \\ \vdots \\ \boldsymbol{\psi}_{\boldsymbol{P}_r}(\boldsymbol{k}_t) \end{bmatrix}, \qquad \boldsymbol{y}_t = \begin{bmatrix} \boldsymbol{q}_t \\ \alpha_1 \boldsymbol{\psi}_{\boldsymbol{P}_1}(\boldsymbol{q}_t) \\ \vdots \\ \alpha_r \boldsymbol{\psi}_{\boldsymbol{P}_r}(\boldsymbol{q}_t) \end{bmatrix}. \tag{9}$$

The corresponding fast coefficient states are collected into

$$\boldsymbol{H}_t = \begin{bmatrix} \boldsymbol{S}_t & \boldsymbol{R}_t^{(1)} & \cdots & \boldsymbol{R}_t^{(r)} \end{bmatrix}. \tag{10}$$

With $D = (r+1)d$, we have $\boldsymbol{x}_t, \boldsymbol{y}_t \in \mathbb{R}^D$ and $\boldsymbol{H}_t \in \mathbb{R}^{d_v \times D}$. The write address uses the full nonlinear features, whereas α_h appears only in the read address, decoupling online coefficient fitting from the final readout gain.

For Gated DeltaNet, the native matrix state follows

$$\boldsymbol{S}_t = \rho_t \boldsymbol{S}_{t-1}\left(\boldsymbol{I}_d - \beta_t \boldsymbol{k}_t \boldsymbol{k}_t^\top\right) + \beta_t \boldsymbol{v}_t \boldsymbol{k}_t^\top. \tag{11}$$

FVLA applies the same update structure in the expanded response space:

$$\boldsymbol{H}_t = \rho_t \boldsymbol{H}_{t-1}\left(\boldsymbol{I}_D - \eta_t \boldsymbol{x}_t \boldsymbol{x}_t^\top\right) + \eta_t \boldsymbol{v}_t \boldsymbol{x}_t^\top, \qquad \eta_t = \frac{\beta_t}{\max(1, \|\boldsymbol{x}_t\|_2^2)}, \tag{12}$$

where $D = (r+1)d$. Equivalently,

$$\boldsymbol{H}_t = \rho_t \boldsymbol{H}_{t-1} + \eta_t\big(\boldsymbol{v}_t - \rho_t \boldsymbol{H}_{t-1}\boldsymbol{x}_t\big)\boldsymbol{x}_t^\top.$$

The prediction $\rho_t \boldsymbol{H}_{t-1}\boldsymbol{x}_t$ contains contributions from both the linear state $\boldsymbol{S}_t$ and all nonlinear coefficient states $\boldsymbol{R}_t^{(h)}$. They therefore form a single expanded Key-to-Value predictor and are corrected by the same prediction error in one update. We refer to this shared-prediction, shared-error update as the *joint update*.

The final readout is simply $\boldsymbol{o}_t = \boldsymbol{H}_t \boldsymbol{y}_t$, recovering Equation 8. FVLA thus expands the response coordinates while retaining the host's matrix-state update structure. Other matrix-state instantiations are given in Appendix C.

Normalized writing. Feature expansion changes the norm of the write address. The factor $\max(1, \|\boldsymbol{x}_t\|_2^2)$ prevents the added nonlinear features from increasing the effective Delta correction solely through a larger address norm, preserving the scale interpretation of β_t. Appendix C.1 gives the derivation.

3.3 State–Compute Scaling and Chunkwise Computation

For Gated DeltaNet (Yang et al., 2024a), FVLA preserves the scalar-retention and rank-one prediction–correction structure of the original recurrence, replacing only the address and matrix state with their expanded counterparts $\boldsymbol{x}_t$ and $\boldsymbol{H}_t$. The same WY-style chunkwise computation (Yang et al., 2024c;a) therefore applies to the expanded state, avoiding token-by-token materialization of the recurrent update during training. The full recurrent and chunkwise algorithms are provided in Appendix G.

The cost of this expansion is controlled directly by the number of response modes r. Each mode adds one $d_v \times d$ fast state and one dense response matrix $\boldsymbol{P}_h \in \mathbb{R}^{d \times d}$. Excluding projections and feed-forward components shared with the host model, FVLA uses $\mathcal{O}((r+1)dd_v)$ fast-state capacity, $\mathcal{O}(rd^2)$ additional slow parameters, and $\mathcal{O}\big(T[rd^2 + (r+1)dd_v]\big)$ computation. For fixed r, computation remains linear in sequence length and the persistent recurrent state remains independent of sequence length.

4 EXPERIMENTS

4.1 EXPERIMENTAL SETUP

Models. We evaluate FVLA on Gated DeltaNet (GDN) (Yang et al., 2024a) and Kimi Delta Attention (KDA) (Team et al., 2025), using $r \in \{1, 4\}$ response modes. All models trained from scratch use a context length of 2,048.

Baselines. Our primary comparisons are the corresponding GDN and KDA hosts. We additionally reevaluate publicly released checkpoints of Transformer++ (Du et al., 2025), GLA (Yang et al., 2024b), MoM-GDN (Du et al., 2025), and HOLA (Cui, 2026) under the same downstream evaluation protocol.

Model scale. We consider two configuration families: a 16-layer model with hidden size 768 (140M scale) and a 24-layer model with hidden size 1,024 (340M scale). FVLA modifies only the memory module and leaves the remaining host architecture unchanged.

Training. We follow the training recipe of Preconditioned DeltaNet (Tumma et al., 2026). All models trained from scratch use SlimPajama (Soboleva et al., 2023) with the Mistral tokenizer (Jiang et al., 2023). The 140M- and 340M-scale configurations are trained on approximately 5B and 15.7B target tokens, respectively. Paired host and FVLA runs use the same data order, training budget, initialization seed, and optimization configuration. Detailed configurations, parameter counts, training settings, and evaluation protocols are provided in Appendix D.1.

4.2 LANGUAGE MODELING AND IN-CONTEXT RETRIEVAL

To assess how effectively a finite-state model preserves and recovers contextual information, we follow the recall-intensive evaluation setting of BASED (Arora et al., 2024). Our retrieval suite includes FDA, SWDE, and SQuAD completion (Arora et al., 2024), together with DROP (Dua et al., 2019), TriviaQA (Joshi et al., 2017), and NQ-2048 (Kwiatkowski et al., 2019). These tasks emphasize context-dependent recovery of information and therefore provide a complementary test of contextual memory in finite-state sequence models. Alongside retrieval, we report WikiText-2 perplexity (Merity et al., 2017) and five-task zero-shot commonsense accuracy (CS5) (Bisk et al., 2020; Zellers et al., 2019; Sakaguchi et al., 2021; Clark et al., 2018) to assess general modeling quality. Detailed evaluation protocols are provided in Appendix D.2.

Table 1: **Language modeling, commonsense, and in-context retrieval.** All scores except Wiki perplexity are percentages. Avg_4 averages SWDE, SQuAD, DROP, and TriviaQA, excluding FDA and NQ because their evaluated contexts exceed the 2K training window; Avg_6 averages all six retrieval tasks. Bold marks the best result within each host–scale group; † denotes reevaluated public checkpoints.

Model	Wiki↓	CS5↑	**In-context retrieval** (acc) ↑							
	ppl.	acc.	FDA	SWDE	SQuAD	DROP	TQA	NQ	Avg_4	Avg_6
140M-scale / 5B training tokens										
GDN	45.50	40.17	11.43	9.72	19.97	13.13	29.68	6.34	18.13	15.05
+ FVLA ($r = 1$)	45.36	39.96	13.97	15.21	22.39	13.85	32.58	6.15	21.01	17.36
+ FVLA ($r = 4$)	**43.60**	**40.41**	**15.97**	**19.53**	**24.60**	**13.94**	**33.77**	**8.17**	**22.96**	**19.33**
KDA	41.44	40.60	7.71	16.65	23.19	15.19	35.78	7.03	22.70	17.59
+ FVLA ($r = 1$)	41.37	40.66	17.51	15.30	24.20	15.09	36.91	7.95	22.88	19.49
+ FVLA ($r = 4$)	**39.68**	**41.11**	**21.69**	**23.49**	**27.65**	**16.15**	**38.80**	**9.91**	**26.52**	**22.95**
340M-scale / 15.7B training tokens										
GDN	27.38	44.07	24.32	21.60	29.36	19.21	42.65	12.64	28.21	24.96
+ FVLA ($r = 1$)	26.76	**44.39**	29.76	28.08	30.16	19.31	43.48	14.22	30.26	27.50
+ FVLA ($r = 4$)	**26.21**	44.05	**35.30**	**30.42**	**32.98**	**19.74**	**44.19**	**16.28**	**31.83**	**29.82**
KDA	26.43	**45.19**	21.42	28.17	28.82	16.09	**45.85**	13.02	29.73	25.56
+ FVLA ($r = 1$)	26.63	44.85	21.69	28.62	29.29	19.31	44.37	15.24	30.40	26.42
+ FVLA ($r = 4$)	**26.01**	44.39	**32.67**	**35.46**	**30.76**	**19.50**	**45.85**	**16.31**	**32.89**	**30.09**
340M-scale reference models / ∼15B training tokens										
Transformer++†	27.03	44.14	28.58	42.84	35.62	21.56	46.03	1.90	36.51	29.42
GLA†	30.36	43.35	6.81	18.00	28.25	17.68	40.76	10.45	26.17	20.33
MoM-GDN†	26.08	43.91	28.68	26.82	29.89	18.30	43.54	12.20	29.64	26.57
HOLA†	30.44	44.20	13.43	16.20	21.08	17.87	44.37	12.54	24.88	20.92

Table 1 shows that FVLA with $r = 4$ improves language modeling and in-context retrieval across hosts and scales. Across four paired comparisons, WikiText-2 perplexity decreases by 1.6–4.3%, while Avg_6 retrieval improves by 17.7–30.5%. CS5 remains stable, with small changes in both directions. With $r = 1$, gains are less consistent across individual metrics, but Avg_6 retrieval still improves in all four settings, with a median relative gain of 10.5%. The more consistent gains at $r = 4$, together with stable CS5, indicate that additional response capacity benefits language modeling and retrieval. Appendix F provides a complementary spectral view, showing a dominant leading direction together with substantial energy across additional directions.

4.3 What Drives the Gain? Response-Space Ablations

We use a separate controlled setting to probe the roles of write normalization, additional recurrent state, and the response functions represented by that state. All variants use the same training conditions and are trained to the same fixed horizon of approximately 45 target tokens per parameter. Expanded-state comparisons use the same $2\times$ main recurrent-state budget, with closely matched trainable parameter counts. For response-augmented variants, we write the added branch compactly as $\boldsymbol{x}_t = [\boldsymbol{k}_t; \boldsymbol{\phi}_{\text{W}}(\boldsymbol{k}_t)]$ and $\boldsymbol{y}_t = [\boldsymbol{q}_t; \boldsymbol{\phi}_{\text{R}}(\boldsymbol{q}_t)]$. Table 2 specifies the corresponding write and read features.

Here $\boldsymbol{\psi}_{\boldsymbol{P}}(\boldsymbol{u}) = c[\boldsymbol{u} \odot (\boldsymbol{P}\boldsymbol{u})]$ denotes the FVLA response feature, $\boldsymbol{P}_0$ fixed random response directions, and $\boldsymbol{\ell}(\boldsymbol{u})$ the learned linear-response control. GDN + Norm and Wide GDN respectively control for write normalization and additional linear-state capacity, while retaining linear responses. Joint Linear then tests whether the jointly updated expanded state helps without nonlinear responses. FVLA (Random) introduces nonlinear responses with fixed directions, FVLA ($\alpha = 1$) learns the response directions but fixes the readout gain, and full FVLA learns both. These comparisons isolate three factors behind the gain: **state allocation, nonlinear response structure ($\boldsymbol{\psi}$), and readout calibration (α).**

Table 2: **Controlled response-space ablations.** $\boldsymbol{\phi}_{\text{W}}$ and $\boldsymbol{\phi}_{\text{R}}$ denote added write/read features; $2\times$ variants share the same state budget. Avg_5 averages FDA, SWDE, SQuAD, DROP, and TriviaQA (%). See Appendix D.3 for full results.

Variant	$\boldsymbol{\phi}_{\text{W}}$	$\boldsymbol{\phi}_{\text{R}}$	State	NLL ↓	Avg_5 ↑
GDN	–	–	$1\times$	3.7047	5.22
GDN + Norm	–	–	$1\times$	3.6851	5.03
Wide GDN	–	–	$2\times$	3.7129	5.15
Joint Linear	$\boldsymbol{\ell}$	$\alpha\boldsymbol{\ell}$	$2\times$	3.6861	5.31
FVLA (Random)	$\boldsymbol{\psi}_{\boldsymbol{P}_0}$	$\alpha\boldsymbol{\psi}_{\boldsymbol{P}_0}$	$2\times$	3.6585	5.72
FVLA ($\alpha = 1$)	$\boldsymbol{\psi}_{\boldsymbol{P}}$	$\boldsymbol{\psi}_{\boldsymbol{P}}$	$2\times$	3.6862	5.68
FVLA (ours)	$\boldsymbol{\psi}_{\boldsymbol{P}}$	$\alpha\boldsymbol{\psi}_{\boldsymbol{P}}$	$2\times$	**3.6427**	**5.88**

State allocation matters. The linear controls show that additional recurrent state alone does not consistently improve modeling quality. Joint-Linear improves over GDN, whereas allocating the same state budget to a wider native linear memory does not. The benefit therefore depends on what response functions the added state is used to represent.

Nonlinear responses use state more effectively. Under the same expanded-state budget, nonlinear responses with fixed random directions outperform the linear controls in language-modeling NLL and aggregate retrieval accuracy, and learning the shared response directions with a calibrated readout improves both further. This supports the central hypothesis that finite fast-state capacity is more effective when allocated to richer Query-dependent responses than to additional linear state.

Readout calibration further improves modeling. Fixing $\alpha = 1$ ties the nonlinear write and readout scales and leads to higher NLL and lower aggregate retrieval accuracy than the learned-α configuration. Because the linear and nonlinear branches are produced by different features and fast states, their unscaled outputs need not share a common scale. Learning α therefore calibrates their relative contribution, decoupling online coefficient fitting from the strength of the nonlinear response at readout. Appendix E shows that the calibrated nonlinear response remains substantial and that removing it at inference causes a large performance degradation.

4.4 Scaling Response Capacity: Quality–State–Compute Trade-off

The number of response modes r directly controls how much recurrent capacity FVLA allocates to Query-dependent responses. We therefore sweep r under matched training conditions and closely matched trainable-parameter budgets to ask whether increasing response-space capacity continues

to improve modeling quality. As shown in Figure 3, the sweep reveals a consistent quality–resource trade-off: broader response spaces improve modeling over the sweep, while persistent state and operator cost grow predictably with the number of modes.

The full quadratic response achieves the lowest NLL but uses substantially more persistent state. We use $r = 4$ in the main experiments because it improves NLL over lower-r settings while using less state and computation than higher-r settings. Figure 3 shows the resulting quality–resource trade-off.

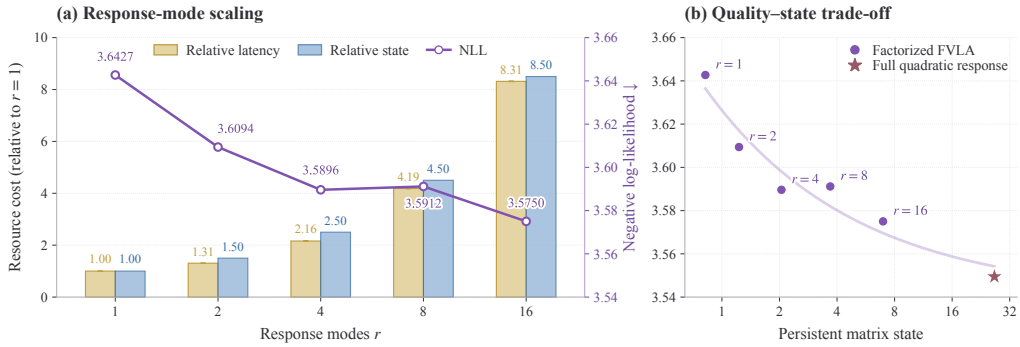


Figure 3: **Response-mode scaling and quality–state trade-off.** a) Confirmation NLL, persistent-state size, and operator latency as the number of response modes r increases. State and latency are normalized to $r = 1$; b) Confirmation NLL versus persistent matrix state for factorized FVLA and the full quadratic response. State is measured in millions of elements on a logarithmic axis.

4.5 Length Extrapolation and Long-Range Retrieval

All models are trained with a 2K context and evaluated beyond the training length without long-context continued training. Because a fixed-size recurrent state must compress an increasing amount of history, we use the S-NIAH tasks from RULER (Hsieh et al., 2024) to test whether expanding the response space improves retrieval from this compressed state.

Figure 4 reveals a directional effect of response-space capacity. Even a single response mode can substantially outperform the host in particular retrieval regimes, showing that a low-dimensional Query-dependent response space already enables associations that are difficult to recover through the host's linear response alone. Increasing to $r = 4$ broadens these gains across retrieval regimes, with the clearest extension appearing in long-range S-NIAH-1 retrieval. This pattern supports interpreting r as controlling the *coverage* of Query-dependent responses: a small response space can specialize to a limited set of retrieval patterns, while additional modes support a broader family of responses from the same compressed history.

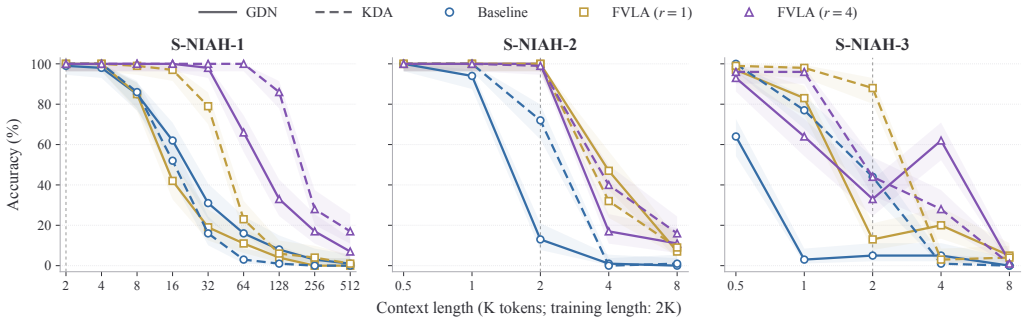


Figure 4: **Length extrapolation on RULER S-NIAH.** 340M models trained with 2K contexts are evaluated without long-context continued training. The vertical gray dashed line marks the training length. Additional model scales are reported in Appendix D.3.

4.6 End-to-End Long-Context Efficiency

We finally measure the end-to-end cost of expanding the response space. Figure 5 separates prefill and decode time and reports peak GPU memory over prompt lengths from 2K to 128K, alongside training throughput at 2K. For $r = 1$, FVLA retains approximately 83% and 78% of the GDN and KDA training throughput, respectively. Its additional decoding cost varies little with prompt length, while preserving the constant-memory scaling of the recurrent state. This modest systems cost accompanies a median relative improvement of 10.5% in six-task average retrieval accuracy across the four host–scale pairs with $r = 1$. Thus, the retrieval gains from response-space expansion do not require sequence-length-dependent memory or a change in the asymptotic inference behavior of the host.

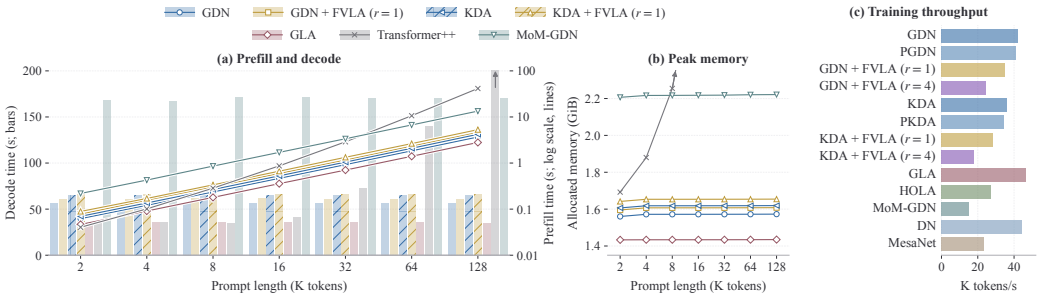


Figure 5: **End-to-end inference and training efficiency.** a) Prefill time (lines, right logarithmic axis) and decode time (bars, left axis) on an RTX 3090, with batch size one, 2K-token prefill chunks, and 1,024 generated tokens. b) Peak allocated GPU memory. RTX 5090 training throughput with 2K sequences and 0.5M tokens per update. All measurements use FP32 parameters with BF16 autocast. GDN/KDA and FVLA use FLA-based implementations (Yang & Zhang, 2024); HOLA, MoM-GDN, and Transformer++ use public checkpoints.

5 Conclusion, Limitations and Future Work

Conclusion. We studied response-space allocation as a complementary design dimension for finite-state associative memory. FVLA augments matrix-state linear attention with structured Query-dependent responses, combining response directions learned across contexts with Value coefficients adapted online within each context. This construction enriches the fixed-state Query-to-Value function while remaining linear in the fast-state coefficients, preserving joint Delta-style updates and chunkwise computation. Across GDN and KDA at two model scales, FVLA consistently improves in-context retrieval and, with four response modes, language modeling performance, while controlled ablations show that allocating additional recurrent capacity to nonlinear responses is more effective than the tested linear state expansions. Together, these results suggest that how recurrent state is used to parameterize the memory response is an important architectural choice alongside how that state is updated.

Limitations and Future Work. Increasing the number of response modes expands both the recurrent address and the persistent state, introducing a non-negligible constant-factor cost even though FVLA preserves the host's linear sequence complexity and fixed-size recurrent memory. A key direction is therefore to retain the benefits of larger response spaces while reducing their computational footprint, through more compact response representations, more efficient quadratic feature computation, and optimized operators for the expanded recurrence. Our experiments currently scale to 340M parameters; extending FVLA to larger models and longer training budgets is another important direction.

AI use statement

We used generative AI tools for language polishing, code review and reorganization, figure preparation, literature search, and assistance with selected mathematical proofs. AI tools were not used to

generate the research ideas, design the experiments, or draft the core scientific narrative of the paper. The authors take full responsibility for the final content of this work, including all AI-assisted contributions.

Reproducibility statement

Our code is released through an anonymous repository at `https://anonymous.4open.science/r/FVLA-61E8` and all models trained from scratch will be released on Hugging Face upon de-anonymization. We provide the training and evaluation scripts to make sure that all the results in the paper can be easily reproduced. Additional implementation and evaluation details are provided in Appendix D.

References

Hanno Ackermann, Hong Cai, Mohsen Ghafoorian, and Amir Habibian. HLA: Hadamard linear attention. In *The 3rd Workshop on Efficient and On-Device Generation (EDGE)*, pp. 1–6, 2026. URL https://openreview.net/forum?id=J8CZ0KH6k3.

Simran Arora, Sabri Eyuboglu, Michael Zhang, Aman Timalsina, Silas Alberti, James Zou, Atri Rudra, and Christopher Re. Simple linear attention language models balance the recall-throughput tradeoff. In *Forty-First International Conference on Machine Learning*, June 2024.

Ali Behrouz, Zeman Li, Yuan Deng, Peilin Zhong, Meisam Razaviyayn, and Vahab Mirrokni. Memory caching: RNNs with growing memory. In *Forty-Third International Conference on Machine Learning*, June 2026a.

Ali Behrouz, Zeman Li, Praneeth Kacham, Majid Daliri, Yuan Deng, Peilin Zhong, Meisam Razaviyayn, and Vahab Mirrokni. ATLAS: Learning to optimally memorize the context at test time. In *Forty-Third International Conference on Machine Learning*, June 2026b.

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. Piqa: Reasoning about physical commonsense in natural language. In *Thirty-Fourth AAAI Conference on Artificial Intelligence*, 2020.

Tommaso Cerruti, Tim Rieder, George Rowlands, Lingfeng Jin, and Imanol Schlag. Linear attention architectures: mechanisms, trade-offs, and cross-layer routing, July 2026. URL http://arxiv.org/abs/2607.07953.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv:1803.05457v1*, 2018.

Wanyun Cui. A hippocampus for linear attention: an exact memory for what the recurrent state forgets, July 2026. URL http://arxiv.org/abs/2607.02303.

Tri Dao and Albert Gu. Transformers are SSMs: Generalized models and efficient algorithms through structured state space duality. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *ICML'24*, pp. 10041–10071, Vienna, Austria, July 2024. JMLR.org.

Jusen Du, Weigao Sun, Disen Lan, Jiaxi Hu, Tao Zhang, and Yu Cheng. MoM: Linear sequence modeling with mixture-of-memories. In *The Fourteenth International Conference on Learning Representations*, October 2025.

Dheeru Dua, Yizhong Wang, Pradeep Dasigi, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. DROP: A reading comprehension benchmark requiring discrete reasoning over paragraphs. In *Proc. of NAACL*, 2019.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. The language model evaluation harness, 07 2024. URL https://zenodo.org/records/12608602.

Ali Hatamizadeh, Yejin Choi, and Jan Kautz. Gated DeltaNet-2: decoupling erase and write in linear attention, May 2026. URL http://arxiv.org/abs/2605.22791.

Cheng-Ping Hsieh, Simeng Sun, Samuel Kriman, Shantanu Acharya, Dima Rekesh, Fei Jia, and Boris Ginsburg. RULER: What's the real context size of your long-context language models? In *First Conference on Language Modeling*, August 2024.

Jiaxi Hu, Yongqi Pan, Jusen Du, Disen Lan, Xiaqiang Tang, Qingsong Wen, Yuxuan Liang, and Weigao Sun. Improving bilinear RNN with closed-loop control. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, October 2025.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mistral 7B, October 2023.

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

Praneeth Kacham, Vahab Mirrokni, and Peilin Zhong. PolySketchFormer: fast transformers via sketching polynomial kernels. In *Proceedings of the 41st International Conference on Machine Learning*, pp. 22748–22770. PMLR, July 2024. URL https://proceedings.mlr.press/v235/kacham24a.html.

Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are RNNs: Fast autoregressive transformers with linear attention. In *Proceedings of the 37th International Conference on Machine Learning*, pp. 5156–5165. PMLR, November 2020.

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

Aakash Lahoti, Kevin Li, Berlin Chen, Caitlin Wang, Aviv Bick, J. Zico Kolter, Tri Dao, and Albert Gu. Mamba-3: Improved sequence modeling using state space principles. In *The Fourteenth International Conference on Learning Representations*, October 2025.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. In *International Conference on Learning Representations*, February 2017.

E. A. Nadaraya. On estimating regression. *Theory of Probability & Its Applications*, 9(1):141–142, 1964. doi: 10.1137/1109020.

Zongyue Qin, Haoran Deng, Jason Cong, and Yizhou Sun. Learnable low-rank polynomial sketch for effective linear attention. In *ICML'26 Workshop on CoLoRAI - The 2nd Workshop on Connecting Low-rank Representations in AI*, May 2026.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. WinoGrande: An adversarial winograd schema challenge at scale. *Communications of the ACM*, 64(9):99–106, September 2021. ISSN 0001-0782, 1557-7317. doi: 10.1145/3474381.

Imanol Schlag, Kazuki Irie, and Jürgen Schmidhuber. Linear transformers are secretly fast weight programmers. In Marina Meila and Tong Zhang (eds.), *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 9355–9366. PMLR, July 2021.

Daria Soboleva, Faisal Al-Khateeb, Robert Myers, Jacob R Steeves, Joel Hestness, and Nolan Dey. SlimPajama: A 627B token cleaned and deduplicated version of RedPajama, 6 2023. URL https://huggingface.co/datasets/cerebras/SlimPajama-627B.

Yu Sun, Xinhao Li, Karan Dalal, Jiarui Xu, Arjun Vikram, Genghan Zhang, Yann Dubois, Xinlei Chen, Xiaolong Wang, Sanmi Koyejo, Tatsunori Hashimoto, and Carlos Guestrin. Learning to (learn at test time): RNNs with expressive hidden states. In *Forty-Second International Conference on Machine Learning*, June 2025.

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*, abs/2307.08621, 2023.

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, November 2025. URL http://arxiv.org/abs/2510.26692.

Neehal Tumma, Noel Loo, and Daniela Rus. Preconditioned DeltaNet: Curvature-aware sequence modeling for linear recurrences. In *Forty-Third International Conference on Machine Learning*, June 2026.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

Johannes von Oswald, Nino Scherrer, Seijin Kobayashi, Luca Versari, Songlin Yang, Maximilian Schlegel, Kaitlin Maile, Yanick Schimpf, Oliver Sieberling, Alexander Meulemans, Guillaume Lajoie, Rif A. Saurous, Charlotte Frenkel, Razvan Pascanu, Blaise Aguera y Arcas, and Joao Sacramento. MesaNet: Sequence modeling by locally optimal test-time training. In *The Fourteenth International Conference on Learning Representations*, October 2025.

Ke Alexander Wang, Jiaxin Shi, and Emily B. Fox. Test-time regression: a unifying framework for designing sequence models with associative memory. *Journal of Machine Learning Research*, 27 (169):1–41, 2026. URL http://jmlr.org/papers/v27/25-0903.html.

Geoffrey S. Watson. Smooth regression analysis. *Sankhyā: the Indian Journal of Statistics, Series a (1961-2002)*, 26(4):359–372, 1964. ISSN 0581572X.

Songlin Yang and Yu Zhang. FLA: A triton-based library for hardware-efficient implementations of linear attention mechanism. https://github.com/fla-org/flash-linear-attention, 2024. GitHub repository.

Songlin Yang, Jan Kautz, and Ali Hatamizadeh. Gated delta networks: Improving Mamba2 with delta rule. In *The Thirteenth International Conference on Learning Representations*, October 2024a.

Songlin Yang, Bailin Wang, Yikang Shen, Rameswar Panda, and Yoon Kim. Gated linear attention transformers with hardware-efficient training. In *Forty-First International Conference on Machine Learning*, June 2024b.

Songlin Yang, Bailin Wang, Yu Zhang, Yikang Shen, and Yoon Kim. Parallelizing linear transformers with the delta rule over sequence length. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, November 2024c.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2019.

Yifei Zuo, Yutong Yin, Zhichen Zeng, Ang Li, Banghua Zhu, and Zhaoran Wang. Local linear attention: An optimal interpolation of linear and softmax attention for test-time regression. In *The Fourteenth International Conference on Learning Representations*, October 2025.

A Related Work

Matrix-state linear attention and online associative learning. Linear attention compresses a growing history into a fixed-size matrix state (Katharopoulos et al., 2020), which can also be interpreted as a fast-weight associative learner that is continuously updated within the current context (Schlag et al., 2021). Subsequent work has substantially improved how such finite states are updated. DeltaNet uses prediction errors to correct previously stored key–value associations (Yang et al., 2024c). Gated DeltaNet introduces data-dependent retention on top of Delta-style updates (Yang et al., 2024a), while Kimi Delta Attention develops finer-grained gating over the recurrent state (Team et al., 2025), and Gated DeltaNet-2 further decouples erase and write operations (Hatamizadeh et al., 2026). Preconditioned DeltaNet improves the geometry of online regression through curvature-aware updates (Tumma et al., 2026), whereas MesaNet defines its fast weights through a regularized regression solution over the accumulated context (von Oswald et al., 2025). More broadly, gated linear attention, retentive networks, state-space models, and feedback-based recurrent architectures have explored richer retention and state-transition mechanisms for finite-state sequence modeling (Yang et al., 2024b; Sun et al., 2023; Dao & Gu, 2024; Lahoti et al., 2025; Hu et al., 2025). Related work has also studied how recurrent-state capacity affects associative recall (Arora et al., 2024). These developments improve how a finite state is updated, retained, optimized, or enlarged. For models whose memory readout remains $f_t(\boldsymbol{q}) = \boldsymbol{S}_t\boldsymbol{q}$, however, a fixed history still induces a linear Query-to-Value function. FVLA targets the complementary *response-space* axis: given a limited matrix-scale fast-state budget, what Query-dependent responses should that state represent?

Nonlinear fast learning and higher-order responses. A complementary line of work increases the expressive power of efficient sequence models through richer fast learners or higher-order representations. TTT-MLP replaces a linear fast learner with a nonlinear test-time learner (Sun et al., 2025), while ATLAS increases both test-time memory capacity and the sophistication of its inner optimization (Behrouz et al., 2026b). Test-Time Regression explicitly treats the regressor class as a design axis alongside weighting and optimization (Wang et al., 2026), and Local Linear Attention develops a richer local regression estimator within this framework (Zuo et al., 2025). These approaches demonstrate the benefit of more expressive test-time predictors, but obtain this flexibility through a more complex learner or regression procedure.

Feature-based methods instead enrich the representation on which the online learner operates. DPFP introduces product features (Schlag et al., 2021), PolySketchFormer uses polynomial sketches to capture higher-order attention interactions (Kacham et al., 2024), and learnable low-rank polynomial sketches compress larger polynomial feature spaces (Qin et al., 2026). Hadamard Linear Attention similarly introduces higher-order Query–Key interactions through Hadamard products of similarity matrices (Ackermann et al., 2026). While these methods improve nonlinear expressivity, the underlying higher-order spaces grow rapidly with feature order and dimension, making the allocation of representation, state, and computation increasingly important. What remains less explored is how to allocate a limited matrix-scale online-state budget directly to richer Query-dependent memory responses while retaining the simple recurrent update structure of linear attention. FVLA instead focuses on response-space allocation as a complementary design axis, enriching memory responses without replacing existing update and retention mechanisms.

Explicit memory. Another line of work alleviates information loss from recurrent compression by retaining part of the history outside the compressed state. Memory Caching preserves selected recurrent states so that effective memory can grow with context (Behrouz et al., 2026a), while HOLA augments linear attention with a bounded exact key–value memory and uses recurrent prediction error to identify associations worth preserving explicitly (Cui, 2026). Such methods allocate additional capacity to storing particular historical items. FVLA instead changes the response function represented by information already compressed into a fixed-size recurrent state. Explicit memory

therefore addresses which information remains directly accessible, whereas FVLA addresses how compressed information can be reinterpreted as a Query-dependent response.

B Notation and Matrix-State Convention

Unless stated otherwise, all vectors are column vectors. Scalar quantities, including dimensions and normalizers such as D and N_t, are italic; Latin-letter vectors use the template's bold lowercase notation (e.g., $\boldsymbol{q}$), and matrices use bold uppercase notation (e.g., $\boldsymbol{S}$). Vector-valued response features are also bold (e.g., $\boldsymbol{\psi}_{\boldsymbol{P}_h}$). We use t for the sequence position, i for the response-coordinate index, and $h \in \{1, \dots, r\}$ for the response-mode index. The attention-head index is omitted unless needed explicitly. We denote the single-head Key/Query and Value dimensions by d and d_v, respectively, and use r for the number of FVLA response modes. The expanded address dimension is $D = (r+1)d$. Table 3 summarizes the notation used throughout the paper.

Table 3: **Notation used in FVLA.** Fast-state quantities depend on the current context and sequence position, whereas $\boldsymbol{P}_h$, c_h, and α_h are learned by the outer language-modeling objective.

Symbol	Dimension	Meaning
t	–	Sequence position
i	–	Response-coordinate index
h	–	Response-mode index
r	–	Number of response modes
d, d_v	–	Key/Query and Value dimensions
$D = (r+1)d$	–	Expanded FVLA address dimension
$\boldsymbol{q}_t, \boldsymbol{k}_t$	$\mathbb{R}^d$	Query and Key
$\boldsymbol{v}_t, \boldsymbol{o}_t, \boldsymbol{e}_t$	$\mathbb{R}^{d_v}$	Value, output, and prediction error
$\boldsymbol{S}_t$	$\mathbb{R}^{d_v \times d}$	Standard matrix fast state
$\boldsymbol{s}_{t,i}$	$\mathbb{R}^{d_v}$	Value vector of response coordinate i
$\boldsymbol{C}_{t,i}$	$\mathbb{R}^{d_v \times d}$	Query-dependent response matrix of coordinate i
$\boldsymbol{P}_h$	$\mathbb{R}^{d \times d}$	Shared response-direction matrix of mode h
$\boldsymbol{p}_i^{(h)}$	$\mathbb{R}^d$	Query-modulation direction of coordinate i in mode h
$\boldsymbol{R}_t^{(h)}$	$\mathbb{R}^{d_v \times d}$	Fast coefficient state of response mode h
$\boldsymbol{r}_{t,i}^{(h)}$	$\mathbb{R}^{d_v}$	Online Value coefficient of coordinate i in mode h
$\boldsymbol{\psi}_{\boldsymbol{P}_h}$	$\mathbb{R}^d \to \mathbb{R}^d$	Nonlinear response feature map of mode h
$\boldsymbol{x}_t, \boldsymbol{y}_t$	$\mathbb{R}^D$	Expanded write and read addresses
$\boldsymbol{H}_t$	$\mathbb{R}^{d_v \times D}$	Joint FVLA fast state
c_h	–	Feature scale of response mode h
α_h	–	Readout calibration of response mode h
β_t	–	Write gate or update strength
ρ_t	–	Retention factor
N_t	–	Write-normalization factor

For host-specific gates, g_t denotes a scalar log-retention in GDN, while $\boldsymbol{g}_t \in \mathbb{R}^d$ and $\boldsymbol{g}_t^F \in \mathbb{R}^D$ denote the native and expanded log-retention vectors in KDA. The scalar ρ_t denotes a retention factor in the abstract GDN recurrence.

Matrix-state orientation. Throughout the paper, we use a *Value-by-Address* convention for matrix fast states. A standard state is therefore written as $\boldsymbol{S}_t \in \mathbb{R}^{d_v \times d}$, with readout $\boldsymbol{S}_t \boldsymbol{q}_t$ and Key–Value outer-product write $\boldsymbol{v}_t \boldsymbol{k}_t^\top$. Under this convention, $\boldsymbol{S}_t$ can be viewed directly as a map from the Key/Query address space to the Value space, and its i-th column $\boldsymbol{s}_{t,i}$ is the Value vector associated with the i-th response coordinate. This orientation is also convenient for the function-valued interpretation in Section 3: the Query controls how response coordinates are addressed, while their coefficients live in the Value dimension.

Some linear-attention formulations store the transposed state, $\widetilde{\boldsymbol{S}}_t \in \mathbb{R}^{d \times d_v}$, and write the corresponding readout as $\widetilde{\boldsymbol{S}}_t^\top \boldsymbol{q}_t$. We uniformly convert such expressions by setting $\boldsymbol{S}_t = \widetilde{\boldsymbol{S}}_t^\top$. The two

notational conventions are related by transposition; all recurrences in this paper, including the host extensions in Appendix C, are written in the Value-by-Address orientation.

FVLA states and addresses. The same convention extends directly to FVLA. For response mode h, $\boldsymbol{P}_h$ contains the response directions learned across contexts, whereas $\boldsymbol{R}_t^{(h)}$ contains their context-dependent Value coefficients. The nonlinear feature map $\boldsymbol{\psi}_{\boldsymbol{P}_h} : \mathbb{R}^d \to \mathbb{R}^d$ therefore produces an address-space feature, which is mapped to the Value space by $\boldsymbol{R}_t^{(h)}$.

The linear state and all response-mode states are concatenated along the address dimension,

$$\boldsymbol{H}_t = \begin{bmatrix} \boldsymbol{S}_t & \boldsymbol{R}_t^{(1)} & \cdots & \boldsymbol{R}_t^{(r)} \end{bmatrix} \in \mathbb{R}^{d_v \times D}, \qquad \boldsymbol{o}_t = \boldsymbol{H}_t \boldsymbol{y}_t. \tag{13}$$

Accordingly, $\boldsymbol{x}_t$ and $\boldsymbol{y}_t$ denote the expanded write and read addresses in $\mathbb{R}^D$. The write address contains the Key and its nonlinear response features, whereas the read address contains the Query features calibrated by the readout gains α_h. This notation lets the linear state and the nonlinear coefficient states be treated uniformly as blocks of a single Value-by-Address fast state in the joint updates used throughout the paper.

C Normalized Writing and Host Extensions

C.1 Normalized Delta Writing in the Expanded Address Space

Gated DeltaNet applies ℓ_2 normalization to the Key, so its native write address satisfies $\|\boldsymbol{k}_t\|_2^2 = 1$. Let $\bar{\boldsymbol{S}}_t$ denote the retained state before the current write, and define the prediction error as $\boldsymbol{e}_t = \boldsymbol{v}_t - \bar{\boldsymbol{S}}_t \boldsymbol{k}_t$. The standard Delta update is

$$\boldsymbol{S}_t = \bar{\boldsymbol{S}}_t + \beta_t \boldsymbol{e}_t \boldsymbol{k}_t^\top. \tag{14}$$

Evaluating the state change at the current Key gives

$$\boldsymbol{S}_t \boldsymbol{k}_t - \bar{\boldsymbol{S}}_t \boldsymbol{k}_t = \beta_t \boldsymbol{e}_t, \tag{15}$$

where we used $\|\boldsymbol{k}_t\|_2^2 = 1$. Thus, in the native Gated Delta update, the write gate β_t directly controls the fraction of the current Key–Value prediction error corrected by this write.

FVLA replaces the original Key address with the expanded write address

$$\boldsymbol{x}_t = \begin{bmatrix} \boldsymbol{k}_t \\ \boldsymbol{\psi}_{\boldsymbol{P}_1}(\boldsymbol{k}_t) \\ \vdots \\ \boldsymbol{\psi}_{\boldsymbol{P}_r}(\boldsymbol{k}_t) \end{bmatrix}. \tag{16}$$

Although the original Key remains normalized, the expanded address is generally not unit norm. In particular,

$$\|\boldsymbol{x}_t\|_2^2 = \|\boldsymbol{k}_t\|_2^2 + \sum_{h=1}^{r} \|\boldsymbol{\psi}_{\boldsymbol{P}_h}(\boldsymbol{k}_t)\|_2^2 = 1 + \sum_{h=1}^{r} \|\boldsymbol{\psi}_{\boldsymbol{P}_h}(\boldsymbol{k}_t)\|_2^2 . \tag{17}$$

If the original Delta rule were applied directly to this expanded address, the joint state would be updated as $\boldsymbol{H}_t = \bar{\boldsymbol{H}}_t + \beta_t \boldsymbol{e}_t \boldsymbol{x}_t^\top$. The resulting correction at the current write address would then be

$$\boldsymbol{H}_t \boldsymbol{x}_t - \bar{\boldsymbol{H}}_t \boldsymbol{x}_t = \beta_t \|\boldsymbol{x}_t\|_2^2 \boldsymbol{e}_t. \tag{18}$$

Consequently, adding response features would increase the effective correction magnitude simply by increasing the norm of the address. The meaning of the original write gate β_t would therefore depend on the number and scale of the added response features.

To preserve the correction scale of the native Delta update, FVLA uses the normalized write

$$\boldsymbol{H}_t = \bar{\boldsymbol{H}}_t + \frac{\beta_t}{N_t} \boldsymbol{e}_t \boldsymbol{x}_t^\top, \qquad N_t = \max\bigl(1, \|\boldsymbol{x}_t\|_2^2\bigr). \tag{19}$$

The corresponding correction at the current address is

$$\boldsymbol{H}_t\boldsymbol{x}_t - \bar{\boldsymbol{H}}_t\boldsymbol{x}_t = \beta_t \frac{\|\boldsymbol{x}_t\|_2^2}{N_t}\boldsymbol{e}_t. \tag{20}$$

For the Gated Delta instances used in this work, $\|\boldsymbol{k}_t\|_2^2 = 1$ implies $\|\boldsymbol{x}_t\|_2^2 \geq 1$, and hence $N_t = \|\boldsymbol{x}_t\|_2^2$. Equation 20 therefore reduces to

$$\boldsymbol{H}_t\boldsymbol{x}_t - \bar{\boldsymbol{H}}_t\boldsymbol{x}_t = \beta_t\boldsymbol{e}_t. \tag{21}$$

The normalized FVLA write thus preserves the scale interpretation of β_t: expanding the response space changes the representation of the write address without increasing the correction applied to the current Key–Value association merely through a larger address norm.

We retain the more general form $N_t = \max(1, \|\boldsymbol{x}_t\|_2^2)$ when writing the FVLA update so that the same rule can also be used with hosts whose write addresses are not guaranteed to have unit norm. The lower bound prevents normalization by a small address norm from amplifying the update.

C.2 Host Recurrences in the Expanded Response Space

FVLA expands the response coordinates of a matrix-state learner without prescribing a particular online update rule. Once the native Key address $\boldsymbol{k}_t$ and matrix state are replaced by the expanded write address $\boldsymbol{x}_t \in \mathbb{R}^D$ and the corresponding fast state, the host can retain its original state-transition structure. Table 4 summarizes this construction for several representative matrix-state recurrences. GDN(Yang et al., 2024a) and KDA(Team et al., 2025) correspond to the hosts evaluated in this work; the remaining rows give algebraic extensions of the same response-space construction.

For consistency, all recurrences are written using the Value-by-Address orientation introduced in Appendix B. We omit short convolutions, normalization layers, output gates, and other components that do not affect the core matrix-state recurrence.

Table 4: **FVLA extensions of representative matrix-state recurrences.** Here $\boldsymbol{x}_t, \boldsymbol{y}_t \in \mathbb{R}^D$ denote the expanded write and read addresses, $D = (r+1)d$, and $\eta_t = \beta_t/N_t$ with $N_t = \max(1, \|\boldsymbol{x}_t\|_2^2)$ for normalized Delta updates.

Method	State transition	Readout
LA	$\boldsymbol{S}_t = \boldsymbol{S}_{t-1} + \boldsymbol{v}_t\boldsymbol{k}_t^\top$	$\boldsymbol{o}_t = \boldsymbol{S}_t\boldsymbol{q}_t$
+FVLA	$\boldsymbol{H}_t = \boldsymbol{H}_{t-1} + \boldsymbol{v}_t\boldsymbol{x}_t^\top$	$\boldsymbol{o}_t = \boldsymbol{H}_t\boldsymbol{y}_t$
Mamba-2	$\boldsymbol{S}_t = a_t\boldsymbol{S}_{t-1} + \boldsymbol{v}_t\boldsymbol{k}_t^\top$	$\boldsymbol{o}_t = \boldsymbol{S}_t\boldsymbol{q}_t$
+FVLA	$\boldsymbol{H}_t = a_t\boldsymbol{H}_{t-1} + \boldsymbol{v}_t\boldsymbol{x}_t^\top$	$\boldsymbol{o}_t = \boldsymbol{H}_t\boldsymbol{y}_t$
DeltaNet	$\boldsymbol{S}_t = \boldsymbol{S}_{t-1}(\boldsymbol{I}_d - \beta_t\boldsymbol{k}_t\boldsymbol{k}_t^\top) + \beta_t\boldsymbol{v}_t\boldsymbol{k}_t^\top$	$\boldsymbol{o}_t = \boldsymbol{S}_t\boldsymbol{q}_t$
+FVLA	$\boldsymbol{H}_t = \boldsymbol{H}_{t-1}(\boldsymbol{I}_D - \eta_t\boldsymbol{x}_t\boldsymbol{x}_t^\top) + \eta_t\boldsymbol{v}_t\boldsymbol{x}_t^\top$	$\boldsymbol{o}_t = \boldsymbol{H}_t\boldsymbol{y}_t$
GDN	$\boldsymbol{S}_t = \rho_t\boldsymbol{S}_{t-1}(\boldsymbol{I}_d - \beta_t\boldsymbol{k}_t\boldsymbol{k}_t^\top) + \beta_t\boldsymbol{v}_t\boldsymbol{k}_t^\top$	$\boldsymbol{o}_t = \boldsymbol{S}_t\boldsymbol{q}_t$
+FVLA	$\boldsymbol{H}_t = \rho_t\boldsymbol{H}_{t-1}(\boldsymbol{I}_D - \eta_t\boldsymbol{x}_t\boldsymbol{x}_t^\top) + \eta_t\boldsymbol{v}_t\boldsymbol{x}_t^\top$	$\boldsymbol{o}_t = \boldsymbol{H}_t\boldsymbol{y}_t$
KDA	$\boldsymbol{S}_t = \boldsymbol{S}_{t-1}\,\mathrm{Diag}(\exp(\boldsymbol{g}_t))(\boldsymbol{I}_d - \beta_t\boldsymbol{k}_t\boldsymbol{k}_t^\top) + \beta_t\boldsymbol{v}_t\boldsymbol{k}_t^\top$	$\boldsymbol{o}_t = \boldsymbol{S}_t\boldsymbol{q}_t$
+FVLA	$\boldsymbol{H}_t = \boldsymbol{H}_{t-1}\,\mathrm{Diag}(\exp(\boldsymbol{g}_t^F))(\boldsymbol{I}_D - \eta_t\boldsymbol{x}_t\boldsymbol{x}_t^\top) + \eta_t\boldsymbol{v}_t\boldsymbol{x}_t^\top$	$\boldsymbol{o}_t = \boldsymbol{H}_t\boldsymbol{y}_t$
PDN	$\boldsymbol{S}_t = \boldsymbol{S}_{t-1}(\boldsymbol{I}_d - \beta_t\boldsymbol{k}_t\widetilde{\boldsymbol{k}}_t^\top) + \beta_t\boldsymbol{v}_t\widetilde{\boldsymbol{k}}_t^\top$	$\boldsymbol{o}_t = \boldsymbol{S}_t\boldsymbol{q}_t$
+FVLA	$\boldsymbol{H}_t = \boldsymbol{H}_{t-1}(\boldsymbol{I}_D - \beta_t\boldsymbol{x}_t\widetilde{\boldsymbol{x}}_t^\top) + \beta_t\boldsymbol{v}_t\widetilde{\boldsymbol{x}}_t^\top$	$\boldsymbol{o}_t = \boldsymbol{H}_t\boldsymbol{y}_t$
MesaNet	$\boldsymbol{G}_t = \gamma_t\boldsymbol{G}_{t-1} + \beta_t\boldsymbol{v}_t\boldsymbol{k}_t^\top$ $\boldsymbol{M}_t = \gamma_t\boldsymbol{M}_{t-1} + \beta_t\boldsymbol{k}_t\boldsymbol{k}_t^\top$	$\boldsymbol{o}_t = \boldsymbol{G}_t(\boldsymbol{M}_t + \boldsymbol{\Lambda})^{-1}\boldsymbol{q}_t$
+FVLA	$\boldsymbol{G}_t^F = \gamma_t\boldsymbol{G}_{t-1}^F + \beta_t\boldsymbol{v}_t\boldsymbol{x}_t^\top$ $\boldsymbol{M}_t^F = \gamma_t\boldsymbol{M}_{t-1}^F + \beta_t\boldsymbol{x}_t\boldsymbol{x}_t^\top$	$\boldsymbol{o}_t = \boldsymbol{G}_t^F(\boldsymbol{M}_t^F + \boldsymbol{\Lambda}_F)^{-1}\boldsymbol{y}_t$

For additive and scalar-decay memories, the extension amounts to replacing the native address with the expanded FVLA address while preserving the original state transition. Delta-style hosts retain their prediction–correction structure in the enlarged address space. In particular, DeltaNet and Gated

DeltaNet replace the native rank-one correction $\boldsymbol{k}_t\boldsymbol{k}_t^\top$ with $\boldsymbol{x}_t\boldsymbol{x}_t^\top$, while the normalized step $\eta_t = \beta_t/N_t$ preserves the write scale derived in Appendix C.1. The recurrent state remains Value-by-Address, with the enlarged address dimension D replacing d.

Kimi Delta Attention(Team et al., 2025) additionally applies coordinate-wise retention to the address dimension. We denote its native log-retention vector by $\boldsymbol{g}_t \in \mathbb{R}^d$ and the corresponding expanded log-retention by $\boldsymbol{g}_t^F = [\boldsymbol{g}_t; \ldots; \boldsymbol{g}_t] \in \mathbb{R}^D$, with $r+1$ copies. The exponential in Table 4 is applied elementwise to obtain the retention factors. The response expansion does not otherwise alter the Delta-style prediction–correction structure.

For Preconditioned DeltaNet(Tumma et al., 2026), $\widetilde{\boldsymbol{k}}_t$ denotes the preconditioned write direction in the native address space, while $\widetilde{\boldsymbol{x}}_t$ denotes its counterpart after response expansion. The preconditioner determines the write direction, whereas FVLA changes the response coordinates on which that update operates.

MesaNet(von Oswald et al., 2025) differs from the single-matrix recurrences above because its regression solution maintains both a Value–address cross-covariance state and an address Gram state. We denote these by $\boldsymbol{G}_t$ and $\boldsymbol{M}_t$ to avoid confusion with the FVLA joint state $\boldsymbol{H}_t$. Under response expansion, $\boldsymbol{G}_t^F \in \mathbb{R}^{d_v \times D}$ stores the expanded Value–address statistics, while $\boldsymbol{M}_t^F \in \mathbb{R}^{D\times D}$ is the corresponding Gram state. Consequently, this extension enlarges the Gram state from $d \times d$ to $D \times D$, in addition to expanding the Value-coefficient state.

C.3 Feature-scale initialization

For a response mode, define the unscaled quadratic feature as

$$\boldsymbol{\phi}_{\boldsymbol{P}}(\boldsymbol{u}) = \boldsymbol{u} \odot (\boldsymbol{P}\boldsymbol{u}).$$

Query and Key vectors are L2-normalized in our implementation, so we use unit-norm inputs as the reference distribution for initialization. For $\boldsymbol{u} \sim \mathrm{Unif}(\mathbb{S}^{d-1})$, the expected feature energy is

$$\mathbb{E}\big[\|\boldsymbol{\phi}_{\boldsymbol{P}}(\boldsymbol{u})\|_2^2\big] = \frac{\|\boldsymbol{P}\|_F^2 + 2\|\operatorname{diag}(\boldsymbol{P})\|_2^2}{d(d+2)}.$$

We initialize each $\boldsymbol{P}_h$ as a random orthogonal matrix and compute this energy from the realized initialization.

For r response modes, the implementation uses

$$\boldsymbol{\Psi}(\boldsymbol{u}) = \frac{c}{\sqrt{r}} \begin{bmatrix} \boldsymbol{\phi}_{\boldsymbol{P}_1}(\boldsymbol{u}) \\ \vdots \\ \boldsymbol{\phi}_{\boldsymbol{P}_r}(\boldsymbol{u}) \end{bmatrix} \qquad c = \exp(s).$$

Let e_h denote the expected energy of mode h and $\bar{e} = r^{-1}\sum_{h=1}^r e_h$. We initialize

$$s_0 = -\frac{1}{2}\log \bar{e} + \log \gamma,$$

which gives

$$\mathbb{E}\big[\|\boldsymbol{\Psi}(\boldsymbol{u})\|_2^2\big] = \gamma^2.$$

Thus, the $1/\sqrt{r}$ normalization keeps the total initial nonlinear feature energy independent of the number of response modes. We use $\gamma = 2$ as a fixed initialization target, while the readout strength is calibrated separately by the learnable coefficient α.

D Experimental Details

D.1 Model and Training Setup

Architecture. Following prior work (Tumma et al., 2026; Yang et al., 2024b; Hu et al., 2025; Yang et al., 2024a), we evaluate FVLA on 2 matrix-state hosts, Gated DeltaNet (GDN) (Yang et al., 2024a) and Kimi Delta Attention (KDA) (Team et al., 2025). The 140M-scale configuration uses 16 layers, hidden dimension 768, 6 attention heads, and a SwiGLU intermediate dimension of 2,048;

the 340M-scale configuration uses 24 layers, hidden dimension 1,024, 8 attention heads, and a SwiGLU intermediate dimension of 2,816. Both use a per-head Query/Key dimension of 128. We train GDN and KDA at both scales, together with FVLA variants using $r = 1$ and $r = 4$. All models use pre-normalized residual blocks with RMSNorm and SwiGLU, tied input and output embeddings over a 32,000-token vocabulary, and width-4 depthwise causal convolutions with SiLU on the Query, Key, and Value projections. GDN applies per-head RMSNorm to the memory output, while KDA additionally uses a sigmoid output gate.

Training hyperparameters. All models trained from scratch use SlimPajama (Soboleva et al., 2023) with the 32K Mistral tokenizer (Jiang et al., 2023) and a training context length of 2,048. Tokenized documents are concatenated into a fixed training stream, with recurrent states reset at sequence boundaries. The 140M- and 340M-scale models are trained for 9,537 and 30,000 optimization steps, corresponding to approximately 5B and 15.7B target tokens, respectively, with an effective batch size of approximately 0.5M tokens per update. We use AdamW with peak learning rate 4×10^{-4}, $\beta = (0.9, 0.95)$, $\epsilon = 10^{-8}$, weight decay 0.01, and gradient clipping at a global norm of 1.0. The learning rate is linearly warmed up for 1,024 steps and then cosine-decayed to 0.1 of its peak value. Training uses FP16/FP32 mixed precision with gradient checkpointing. Paired models use the same data order, training budget, optimization configuration, and random seed of 42.

FVLA configuration. The main experiments use $r \in \{1, 4\}$ response modes. Response matrices are parameterized independently for each layer, head, and mode. The response matrices are initialized as random orthogonal matrices, and the nonlinear feature scale is initialized using the energy calibration described in Appendix C.3, with target RMS norm $\gamma = 2$. The readout coefficient α is initialized to 0.1.

D.2 Evaluation Protocols

Language modeling. We evaluate language modeling on the document-level WikiText-2 raw test set (Merity et al., 2017), containing 62 documents. We compute rolling log-likelihoods with a maximum context length of 2,048 and report word-level perplexity. We apply the standard WikiText detokenization and fixed whitespace normalization used by our evaluation code. Perplexity is computed from the total negative log-likelihood over the test set, normalized by the number of whitespace-delimited words in the original documents.

Commonsense reasoning. We evaluate PIQA (Bisk et al., 2020), HellaSwag (Zellers et al., 2019), WinoGrande (Sakaguchi et al., 2021), ARC-Easy and ARC-Challenge (Clark et al., 2018) using `lm-evaluation-harness` (Gao et al., 2024). PIQA, HellaSwag, and WinoGrande use their validation splits, whereas ARC-Easy and ARC-Challenge use their test splits. All tasks are evaluated zero-shot with a maximum context length of 2,048. We report length-normalized accuracy (`acc_norm`) for PIQA, HellaSwag and ARC-Challenge, and accuracy (`acc`) for WinoGrande and ARC-Easy. CS5 is the unweighted mean of the 5 task metrics, computed before rounding.

In-context retrieval. We evaluate FDA, SWDE, and SQuAD completion (Arora et al., 2024), NQ-2048 (Kwiatkowski et al., 2019), TriviaQA (Joshi et al., 2017), and DROP (Dua et al., 2019) on their validation splits. All tasks use the complete released context with fixed, answer-independent formatting. For FDA and SWDE, we remove duplicated trailing question fields and normalize question formatting; SWDE additionally removes the appended `Summary of information above` section. SQuAD completion uses the released input after trimming surrounding whitespace. DROP and TriviaQA append the question to the full context after removing duplicated trailing questions, with TriviaQA additionally removing the `[PAR]`, `[DOC]`, and `[TLE]` markers. NQ-2048 concatenates the released full context with the question using a fixed formatting rule.

Generation uses zero-shot greedy decoding with a maximum total sequence length of 3,072 for FDA, SWDE, SQuAD completion, TriviaQA and DROP, and 11,008 for NQ-2048, with up to 48 generated tokens and newline termination. We report case-insensitive literal answer-substring accuracy. FDA, SWDE, and SQuAD completion use the provided target string, while NQ-2048, TriviaQA, and DROP count a prediction as correct if it contains any provided non-empty answer string. We report two retrieval averages. Avg_4 is the unweighted mean of SWDE, SQuAD, DROP, and TriviaQA. It excludes FDA and NQ because their evaluated contexts exceed the 2,048-token training window.

Avg_6 is the unweighted mean over all six retrieval tasks and is retained as a complete-suite aggregate. Complete task-level results are reported in Table 5.

Table 5: **Complete results for the main experiments.** We report the individual commonsense and in-context retrieval metrics underlying the aggregate scores in Table 1. Avg_4 averages SWDE, SQuAD, DROP, and TriviaQA, excluding the out-of-training-window FDA and NQ settings; Avg_6 averages all six retrieval tasks. Gray shading marks the Transformer++ attention-based reference.

	LM	Commonsense						In-context Retrieval							
Model	Wiki ↓	PIQA	Hella.	Wino.	ARC-E	ARC-C	CS5 ↑	FDA	SWDE	SQuAD	DROP	TQA	NQ	Avg_4 ↑	Avg_6 ↑
140M-scale / 5B training tokens															
GDN	45.50	60.99	29.17	50.51	38.01	22.18	40.17	11.43	9.72	19.97	13.13	29.68	6.34	18.13	15.05
+ FVLA ($r = 1$)	45.36	59.96	29.23	50.59	38.34	21.67	39.96	13.97	15.21	22.39	13.85	32.58	6.15	21.01	17.36
+ FVLA ($r = 4$)	43.60	59.36	29.57	51.38	38.89	22.87	40.41	15.97	19.53	24.60	13.94	33.77	8.17	22.96	19.33
KDA	41.44	61.15	29.86	50.28	39.35	22.35	40.60	7.71	16.65	23.19	15.19	35.78	7.03	22.70	17.59
+ FVLA ($r = 1$)	41.37	61.15	29.87	49.88	40.74	21.67	40.66	17.51	15.30	24.20	15.09	36.91	7.95	22.88	19.49
+ FVLA ($r = 4$)	39.68	60.12	29.94	51.54	41.92	22.01	41.11	21.69	23.49	27.65	16.15	38.80	9.91	26.52	22.95
340M-scale / 15.7B training tokens															
GDN	27.38	64.25	35.65	50.43	46.38	23.63	44.07	24.32	21.60	29.36	19.21	42.65	12.64	28.21	24.96
+ FVLA ($r = 1$)	26.76	65.40	35.61	50.36	46.09	24.49	44.39	29.76	28.08	30.16	19.31	43.48	14.22	30.26	27.50
+ FVLA ($r = 4$)	26.21	65.07	35.97	49.88	46.46	22.87	44.05	35.30	30.42	32.98	19.74	44.19	16.28	31.83	29.82
KDA	26.43	66.43	36.34	51.38	46.21	25.60	45.19	21.42	28.17	28.82	16.09	45.85	13.02	29.73	25.56
+ FVLA ($r = 1$)	26.63	65.34	36.05	51.93	46.51	24.40	44.85	21.69	28.62	29.29	19.31	44.37	15.24	30.40	26.42
+ FVLA ($r = 4$)	26.01	65.18	36.58	49.96	46.76	23.46	44.39	32.67	35.46	30.76	19.50	45.85	16.31	32.89	30.09
340M-scale reference models / ∼15B training tokens															
Transformer++[†]	27.03	64.74	35.06	51.62	45.79	23.46	44.14	28.58	42.84	35.62	21.56	46.03	1.90	36.51	29.42
GLA[†]	30.36	64.69	33.78	50.43	44.74	23.12	43.35	6.81	18.00	28.25	17.68	40.76	10.45	26.17	20.33
MoM-GDN[†]	26.08	65.40	35.92	49.01	45.92	23.29	43.91	28.68	26.82	29.89	18.30	43.54	12.20	29.64	26.57
HOLA[†]	30.44	64.04	35.06	50.91	47.18	23.81	44.20	13.43	16.20	21.08	17.87	44.37	12.54	24.88	20.92

D.3 Additional Experimental Details

Controlled ablations. We conduct the response-space ablations with approximately 26.82M-parameter models trained for 1.2B target tokens (≈ 44.7 target tokens per parameter) at a context length of 2,048. All variants use the same data order, training recipe, and random seed, and are evaluated using their final checkpoints. Trainable parameter counts are closely matched by adjusting the feed-forward width. Retrieval follows the full-context protocol in Appendix D.2 for FDA, SWDE, SQuAD completion, DROP, and TriviaQA; NQ is not evaluated for these ablations. Complete results are reported in Table 6.

For compactness, define the FVLA quadratic feature and its linear control as

$$\boldsymbol{\psi}_{\boldsymbol{P}}(\boldsymbol{u}) = c[\boldsymbol{u} \odot (\boldsymbol{P}\boldsymbol{u})], \qquad \boldsymbol{\ell}_{\boldsymbol{P}}(\boldsymbol{u}) = c_\ell \boldsymbol{P}\boldsymbol{u}.$$

For variants with an expanded state, we write

$$\boldsymbol{x}_t = \begin{bmatrix} \boldsymbol{k}_t \\ \boldsymbol{\phi}_{\mathrm{W}}(\boldsymbol{k}_t) \end{bmatrix}, \qquad \boldsymbol{y}_t = \begin{bmatrix} \boldsymbol{q}_t \\ \boldsymbol{\phi}_{\mathrm{R}}(\boldsymbol{q}_t) \end{bmatrix}, \qquad \boldsymbol{H}_t = [\boldsymbol{S}_t, \boldsymbol{R}_t]$$

and use the joint update

$$\bar{\boldsymbol{H}}_t = \rho_t \boldsymbol{H}_{t-1}, \qquad N_t = \max(1, \|\boldsymbol{x}_t\|_2^2),$$
$$\boldsymbol{H}_t = \bar{\boldsymbol{H}}_t + \frac{\beta_t}{N_t}(\boldsymbol{v}_t - \bar{\boldsymbol{H}}_t \boldsymbol{x}_t)\boldsymbol{x}_t^\top, \qquad \boldsymbol{o}_t = \boldsymbol{H}_t \boldsymbol{y}_t.$$

We omit the output scaling shared across variants.

GDN. Let $\bar{\boldsymbol{S}}_t = \rho_t \boldsymbol{S}_{t-1}$. The native GDN update and readout are

$$\boldsymbol{S}_t = \bar{\boldsymbol{S}}_t + \beta_t(\boldsymbol{v}_t - \bar{\boldsymbol{S}}_t \boldsymbol{k}_t)\boldsymbol{k}_t^\top, \qquad \boldsymbol{o}_t = \boldsymbol{S}_t \boldsymbol{q}_t.$$

GDN + Norm. This variant retains the native linear state and readout, but uses the FVLA feature energy to normalize the write step:

$$N_t^{\mathrm{quad}} = \max\left(1, \|\boldsymbol{k}_t\|_2^2 + \|\boldsymbol{\psi}_{\boldsymbol{P}}(\boldsymbol{k}_t)\|_2^2\right),$$

Table 6: **Complete response-space ablation results.** Params. reports the exact number of trainable parameters, including token embeddings and excluding fixed buffers. NLL is measured on the held-out confirmation split. Retrieval scores are percentages, and Avg_5 is the unweighted mean of FDA, SWDE, SQuAD, DROP, and TriviaQA, computed before rounding. State denotes main persistent matrix-state capacity relative to native GDN. Gray shading marks the high-capacity full quadratic reference, which is not a matched-budget control.

Model	Params.	State	NLL ↓	FDA ↑	SWDE ↑	SQuAD ↑	DROP ↑	TriviaQA ↑	Avg_5 ↑
GDN	26,824,100	1×	3.7047	0.54	1.53	6.27	8.05	9.72	5.22
GDN + Norm	26,817,750	1×	3.6851	0.36	1.71	6.40	6.66	10.01	5.03
Wide GDN	26,824,100	2×	3.7129	0.36	0.54	6.67	8.34	9.83	5.15
Joint Linear	26,817,800	2×	3.6861	0.27	2.25	6.50	9.10	8.41	5.31
FVLA (Random)	26,814,600	2×	3.6585	0.27	1.44	6.90	8.39	11.61	5.72
FVLA ($\alpha = 1$)	26,817,750	2×	3.6862	0.45	2.34	6.57	8.72	10.31	5.68
FVLA	26,817,800	2×	**3.6427**	0.64	1.62	6.87	8.53	11.73	**5.88**
FVLA-Residual	26,824,200	2×	3.6691	0.36	1.71	5.76	7.91	10.84	5.32
FVLA-Full	26,814,600	65×	3.5495	11.16	6.57	13.74	9.54	16.23	11.45

$$\boldsymbol{S}_t = \bar{\boldsymbol{S}}_t + \frac{\beta_t}{N_t^{\text{quad}}}(\boldsymbol{v}_t - \bar{\boldsymbol{S}}_t \boldsymbol{k}_t)\boldsymbol{k}_t^\top.$$

There is no nonlinear fast state or nonlinear readout; $\boldsymbol{P}$ and c are learned only through the write normalization.

Wide GDN. This control assigns the same $2\times$ main matrix-state capacity as $r = 1$ FVLA to the native linear memory by increasing the Key/Query dimension from 64 to 128 while keeping $d_v = 128$. The native GDN update and readout are otherwise unchanged.

The remaining expanded-state variants all use the joint update above and differ in their added write and read features:

Variant	$\boldsymbol{\phi}_{\text{W}}(\boldsymbol{u})$	$\boldsymbol{\phi}_{\text{R}}(\boldsymbol{u})$	Response directions
Joint Linear	$\boldsymbol{\ell}_{\boldsymbol{P}}(\boldsymbol{u})$	$\alpha\boldsymbol{\ell}_{\boldsymbol{P}}(\boldsymbol{u})$	learned
FVLA (Random)	$\boldsymbol{\psi}_{\boldsymbol{P}_0}(\boldsymbol{u})$	$\alpha\boldsymbol{\psi}_{\boldsymbol{P}_0}(\boldsymbol{u})$	fixed random
FVLA ($\alpha = 1$)	$\boldsymbol{\psi}_{\boldsymbol{P}}(\boldsymbol{u})$	$\boldsymbol{\psi}_{\boldsymbol{P}}(\boldsymbol{u})$	learned
FVLA	$\boldsymbol{\psi}_{\boldsymbol{P}}(\boldsymbol{u})$	$\alpha\boldsymbol{\psi}_{\boldsymbol{P}}(\boldsymbol{u})$	learned

Here $\boldsymbol{P}_0$ is fixed at its random orthogonal initialization. Joint Linear preserves the expanded joint predictor while keeping the fixed-state response linear in the Query. FVLA ($\alpha = 1$) fixes the nonlinear readout gain to one from initialization, whereas the standard FVLA configuration learns α.

Two additional variants are included in the complete results. **FVLA-Residual** retains the native GDN update for $\boldsymbol{S}_t$ and learns $\boldsymbol{R}_t$ separately from the native pre-update prediction residual, rather than forming a single joint predictor. **FVLA-Full** replaces the factorized feature with the explicit quadratic feature $\boldsymbol{\phi}_{\text{full}}(\boldsymbol{u}) = c\ \text{vec}(\boldsymbol{u}\boldsymbol{u}^\top)$ and maintains the corresponding full response state. It is a high-capacity reference rather than a state-matched control.

Length extrapolation. We evaluate paired models trained only with 2,048-token contexts directly at longer sequence lengths, without additional long-context training. Long inputs are processed in 2,048-token streaming chunks while carrying the recurrent state across chunks. For language modeling, models consume a variable-length history and are scored on the final 2,048-token target window. Long-range retrieval is evaluated on RULER S-NIAH-1/2/3 (Hsieh et al., 2024). Complete results across all evaluated sequence lengths are reported in Tables 7, 8, and 9. Figures 6 and 7 visualize the 140M- and 340M-scale results, with GLA, HOLA, MoM-GDN, and Transformer++ reference checkpoints additionally included at 340M scale.

The paired GDN/KDA evaluations use BF16 inference and preserve the recurrent state across 2,048-token chunks. Their language-modeling evaluation uses 32 fixed streams from the held-out confirmation split and reports perplexity on the final 2,048-token target window at each history length.

The public reference checkpoints use FP32 parameters with BF16 autocast and retain their recurrent state or KV cache across chunks. RULER uses 100 examples per task–length cell, greedy decoding with at most 128 generated tokens, and all-reference case-insensitive substring accuracy. S-NIAH-1 is evaluated from 2K through 512K tokens; S-NIAH-2 and S-NIAH-3 are evaluated from 512 through 8K tokens. The shorter RULER regime is shown together in Table 7, while the complete S-NIAH-1 range is separated in Table 9 for readability.

Table 7: **RULER length extrapolation through 8K tokens.** All models are trained with 2K contexts and evaluated without continued training or context truncation.

Model	S-NIAH-1			S-NIAH-2					S-NIAH-3				
	2K	4K	8K	512	1K	2K	4K	8K	512	1K	2K	4K	8K
140M-scale / 5B training tokens													
GDN	100.0	100.0	99.0	100.0	98.0	29.0	6.0	4.0	74.0	3.0	2.0	0.0	0.0
+ FVLA ($r = 1$)	99.0	99.0	100.0	91.0	100.0	27.0	2.0	4.0	70.0	51.0	8.0	0.0	0.0
+ FVLA ($r = 4$)	100.0	100.0	100.0	100.0	90.0	6.0	1.0	5.0	61.0	3.0	1.0	2.0	0.0
KDA	100.0	93.0	78.0	100.0	100.0	14.0	3.0	4.0	92.0	36.0	4.0	2.0	1.0
+ FVLA ($r = 1$)	98.0	94.0	69.0	100.0	95.0	30.0	0.0	0.0	88.0	82.0	40.0	0.0	0.0
+ FVLA ($r = 4$)	100.0	100.0	100.0	100.0	93.0	14.0	1.0	1.0	85.0	73.0	11.0	1.0	0.0
340M-scale / 15.7B training tokens													
GDN	99.0	98.0	85.0	100.0	94.0	13.0	1.0	0.0	64.0	3.0	5.0	5.0	0.0
+ FVLA ($r = 1$)	100.0	100.0	85.0	100.0	100.0	100.0	47.0	7.0	97.0	83.0	13.0	20.0	5.0
+ FVLA ($r = 4$)	100.0	100.0	100.0	100.0	100.0	100.0	17.0	11.0	93.0	64.0	33.0	62.0	2.0
KDA	100.0	100.0	86.0	100.0	100.0	72.0	0.0	1.0	100.0	77.0	44.0	1.0	0.0
+ FVLA ($r = 1$)	100.0	100.0	99.0	100.0	100.0	100.0	32.0	9.0	99.0	98.0	88.0	3.0	4.0
+ FVLA ($r = 4$)	100.0	100.0	100.0	100.0	100.0	99.0	40.0	16.0	96.0	96.0	44.0	28.0	1.0
340M-scale public reference checkpoints													
GLA	20.0	8.0	4.0	100.0	98.0	28.0	1.0	0.0	87.0	63.0	6.0	0.0	0.0
HOLA	100.0	47.0	23.0	100.0	100.0	82.0	0.0	0.0	100.0	100.0	75.0	0.0	0.0
MoM-GDN	99.0	100.0	93.0	100.0	100.0	87.0	6.0	2.0	96.0	89.0	49.0	15.0	1.0
Transformer++	100.0	0.0	0.0	100.0	100.0	100.0	0.0	0.0	99.0	92.0	70.0	0.0	0.0

Table 8: **Streaming language-modeling perplexity under length extrapolation.** We report perplexity on the final 2K target window after streaming each indicated amount of held-out context. Lower is better.

Model	2K	4K	8K	16K	32K	64K	128K	256K	512K
140M-scale / 5B training tokens									
GDN	20.16	17.92	18.37	17.30	16.44	15.36	18.56	19.08	20.87
+ FVLA ($r = 1$)	20.27	17.90	18.24	17.16	16.34	15.26	18.66	19.00	20.90
+ FVLA ($r = 4$)	19.80	17.37	17.58	16.51	15.78	14.71	18.24	18.52	20.35
KDA	18.97	16.79	17.15	16.13	15.46	14.36	17.50	17.88	19.62
+ FVLA ($r = 1$)	19.12	16.78	17.13	16.03	15.41	14.31	17.56	17.90	19.61
+ FVLA ($r = 4$)	18.56	16.33	16.48	15.49	14.85	13.80	17.03	17.50	19.11
340M-scale / 15.7B training tokens									
GDN	13.91	12.40	12.57	11.94	11.38	10.85	13.23	13.27	14.41
+ FVLA ($r = 1$)	13.71	12.16	12.40	11.69	11.23	10.65	12.99	13.05	14.18
+ FVLA ($r = 4$)	13.58	12.05	12.18	11.51	11.09	10.47	12.89	12.92	14.04
KDA	13.56	12.10	12.35	11.68	11.12	10.58	12.90	12.94	14.04
+ FVLA ($r = 1$)	13.65	12.18	12.33	11.66	11.19	10.66	12.98	13.01	14.08
+ FVLA ($r = 4$)	13.47	11.98	12.11	11.45	10.94	10.43	12.77	12.80	13.86
340M-scale public reference checkpoints									
GLA	15.86	14.33	14.79	13.97	13.83	12.95	15.41	15.51	17.04
HOLA	13.70	12.19	13.24	12.42	11.64	11.71	14.51	13.52	15.02
MoM-GDN	13.48	12.00	12.16	11.45	11.04	10.51	12.75	12.77	13.87
Transformer++	13.97	212.07	2332.00	4580.42	7851.53	24459.05	55401.12	OOM	OOM

Table 9: **Ultra-long S-NIAH-1 extrapolation.** We report S-NIAH-1 results from 2K through 512K tokens. A dash denotes an unevaluated length after three consecutive zero-accuracy measurements.

Model	2K	4K	8K	16K	32K	64K	128K	256K	512K
140M-scale / 5B training tokens									
GDN	100.0	100.0	99.0	71.0	21.0	7.0	2.0	1.0	1.0
+ FVLA ($r = 1$)	99.0	99.0	100.0	96.0	91.0	56.0	26.0	11.0	7.0
+ FVLA ($r = 4$)	100.0	100.0	100.0	100.0	100.0	100.0	100.0	97.0	52.0
KDA	100.0	93.0	78.0	55.0	48.0	32.0	23.0	12.0	4.0
+ FVLA ($r = 1$)	98.0	94.0	69.0	45.0	6.0	0.0	0.0	0.0	0.0
+ FVLA ($r = 4$)	100.0	100.0	100.0	100.0	99.0	89.0	68.0	50.0	39.0
340M-scale / 15.7B training tokens									
GDN	99.0	98.0	85.0	62.0	31.0	16.0	8.0	3.0	1.0
+ FVLA ($r = 1$)	100.0	100.0	85.0	42.0	19.0	11.0	4.0	0.0	0.0
+ FVLA ($r = 4$)	100.0	100.0	100.0	100.0	98.0	66.0	33.0	17.0	7.0
KDA	100.0	100.0	86.0	52.0	16.0	3.0	1.0	0.0	0.0
+ FVLA ($r = 1$)	100.0	100.0	99.0	97.0	79.0	23.0	6.0	4.0	1.0
+ FVLA ($r = 4$)	100.0	100.0	100.0	100.0	100.0	100.0	86.0	28.0	17.0
340M-scale public reference checkpoints									
GLA	20.0	8.0	4.0	0.0	0.0	0.0	—	—	—
HOLA	100.0	47.0	23.0	14.0	2.0	2.0	3.0	1.0	1.0
MoM-GDN	99.0	100.0	93.0	71.0	37.0	26.0	16.0	6.0	1.0
Transformer++	100.0	0.0	0.0	0.0	—	—	—	—	—

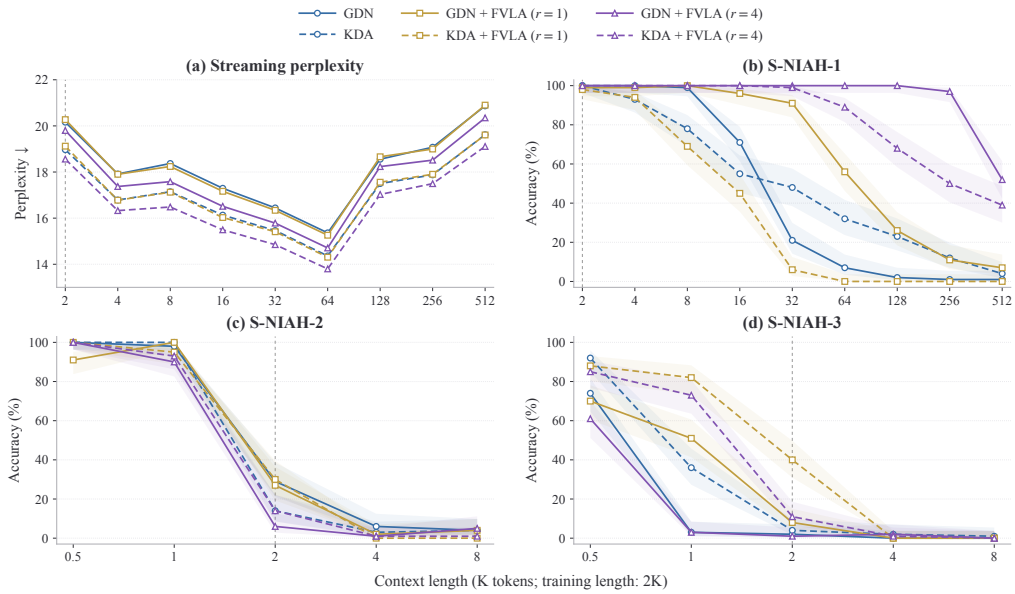


Figure 6: **Length extrapolation at 140M scale.** GDN, KDA, and their FVLA variants are trained on 5B tokens with 2K contexts. a) Perplexity on the final 2K target window of 32 held-out streams. b–d) RULER retrieval accuracy on 100 examples per task and length; shaded bands show 95% Wilson intervals. The vertical gray lines mark the 2K training length.

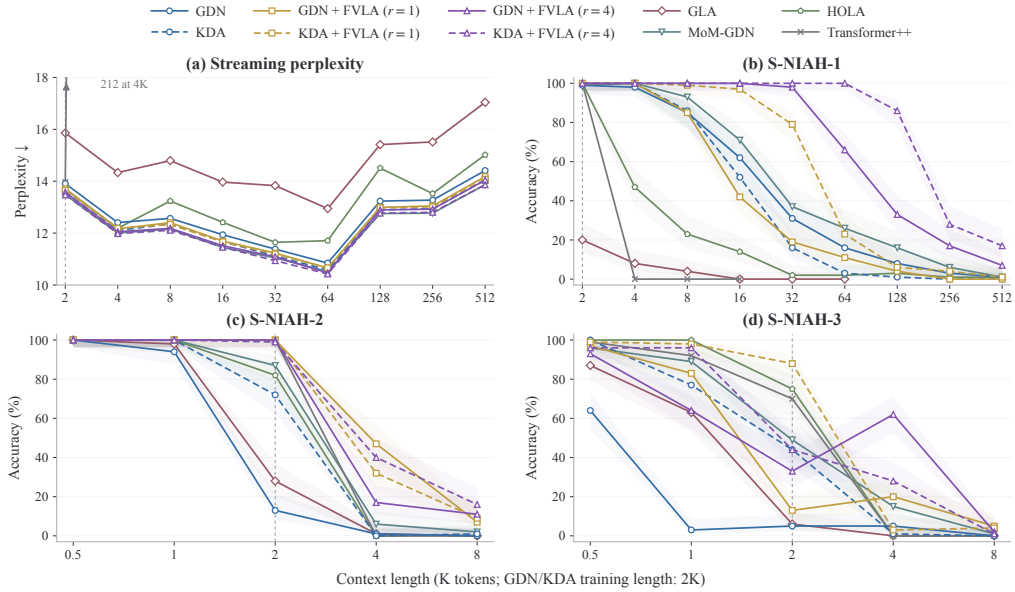


Figure 7: **Length extrapolation at 340M scale with reference models.** The arrow marks Transformer++ perplexity exceeding the displayed range. Retrieval curves end at the last measured point; S-NIAH-1 evaluation for GLA and Transformer++ stops after three consecutive zero-accuracy measurements.

E LEARNED READOUT CALIBRATION IN FVLA

FVLA uses the readout coefficient α to calibrate the relative contribution of the linear and nonlinear responses. Because the 2 branches are produced by different features and fast states, their unscaled output magnitudes need not share a common scale. The coefficient α therefore decouples the scale used to learn the nonlinear fast state from its contribution at readout. We examine this mechanism through the learned α values, the realized response magnitudes, and a runtime intervention on the nonlinear readout.

E.1 LEARNED CALIBRATION AND REALIZED RESPONSE

We first examine whether the trained models make use of the learnable readout coefficient. Figure 8(a) shows the distribution of α across layers and heads for the 4 140M FVLA configurations. The learned values remain nonzero and move away from their common initialization of 0.1, indicating that the nonlinear readout is actively calibrated during training.

The value of α alone does not determine the realized magnitude of the nonlinear response, since the linear and nonlinear branches use different features and fast states. We therefore replay the same readout on fixed unseen contexts and set $\alpha = 0$ only at the Query-side readout, while preserving the nonlinear write path, gating, and resulting fast states. We define the effective nonlinear readout ratio as

$$R_{\text{eff}} = \frac{\sqrt{\sum_t \left\| \boldsymbol{o}_t^{\text{full}} - \boldsymbol{o}_t^{\alpha=0} \right\|_2^2}}{\sqrt{\sum_t \left\| \boldsymbol{o}_t^{\alpha=0} \right\|_2^2}}.$$

The numerator measures the nonlinear contribution that enters the output after readout calibration, while the denominator measures the corresponding linear-only readout magnitude. As shown in Figure 8(b), the typical R_{eff} is close to 1 across the 4 configurations. Thus, learned values of α below 1 do not imply a negligible nonlinear branch: after calibration, the nonlinear and linear responses have comparable realized magnitudes.

E.2 RUNTIME REMOVAL OF THE NONLINEAR READOUT

We next directly measure the trained models' dependence on the nonlinear readout using a fixed-state replay intervention. We first run each checkpoint normally and retain the nonlinear features, gates, and fast states produced on the evaluation sequences. We then replay the readout on the same state trajectory while setting the Query-side α to 0. Thus, the intervention changes only the nonlinear contribution at readout: the write path, gating, and fast states are held fixed to those of the original model. This isolates the contribution of the nonlinear readout from changes in the recurrent state dynamics.

Figure 9 shows a consistent degradation across all 4 140M configurations. Confirmation-set NLL increases by 0.56–0.68, while 2K S-NIAH-1 accuracy decreases by 66–100 percentage points. The nonlinear response is therefore a substantial prediction channel in the trained FVLA models, rather than an inactive component of the expanded fast state. This intervention measures dependence of the trained model on the nonlinear readout.

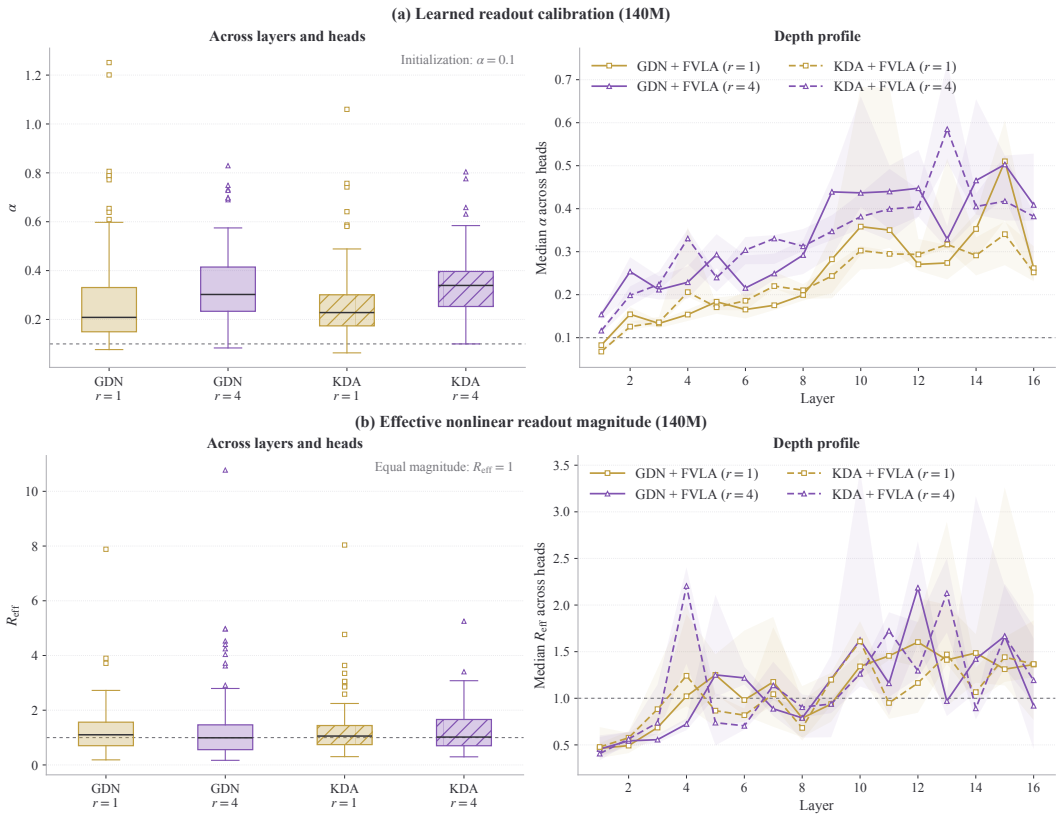


Figure 8: **Learned readout calibration and realized response magnitude.** a) Distribution of α across layers and heads for the 4 140M FVLA configurations; the dashed line marks the initialization value 0.1. b) Effective nonlinear readout ratio R_{eff} on fixed unseen contexts. The replay changes only the Query-side value of α while preserving nonlinear writes and the resulting fast states. The dashed line marks $R_{\text{eff}} = 1$. Shaded bands in the depth profiles show the interquartile range across heads.

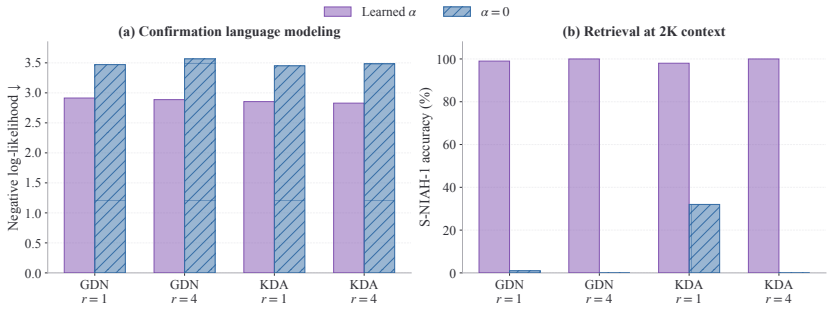


Figure 9: **Effect of removing the nonlinear readout under fixed-state replay.** We compare the same trained checkpoints with their normal readout and with Query-side $\alpha = 0$. Nonlinear writes and fast-state evolution are kept unchanged. Left: NLL on the full confirmation split. Right: 2K S-NIAH-1 accuracy.

F Spectral Analysis of Response Modes

Setup. Each FVLA response mode adds one learned Query-modulation direction per original memory coordinate. To characterize how response capacity is distributed across directions, we analyze the full quadratic reference from the 26.82M-scale response-mode study, trained on 1.2B tokens with a 2K context window and Query dimension $d = 64$. This reference explicitly represents all pairwise Query interactions. We freeze its final checkpoint and collect online states at prefix endpoints from 256 contexts in each of two held-out splits, validation and confirmation. Each split contains 16 contexts at each of 16 prefix lengths from 128 to 2048 tokens.

Measurement. For each original memory coordinate i, we reconstruct the effective quadratic response matrix $\mathbf{C}_{t,i}^{\text{eff}}$, including the readout gain and feature scale. We compute $\mathbf{M}_i = \mathbb{E}_t[(\mathbf{C}_{t,i}^{\text{eff}})^\top \mathbf{C}_{t,i}^{\text{eff}}]$ and denote its eigenvalues in descending order by $\lambda_{i,1}, \ldots, \lambda_{i,d}$. The cumulative coefficient energy is

$$E(k) = \frac{\sum_i \sum_{j=1}^{k} \lambda_{i,j}}{\sum_i \sum_{j=1}^{d} \lambda_{i,j}},$$

where the sums also include all layers and heads. We use k to denote the number of retained spectral directions per coordinate, while r denotes the number of response modes used during training. For the layer-wise analysis, we compute $E(k)$ within each layer and report k_{90}, the smallest k that retains 90% of its coefficient energy.

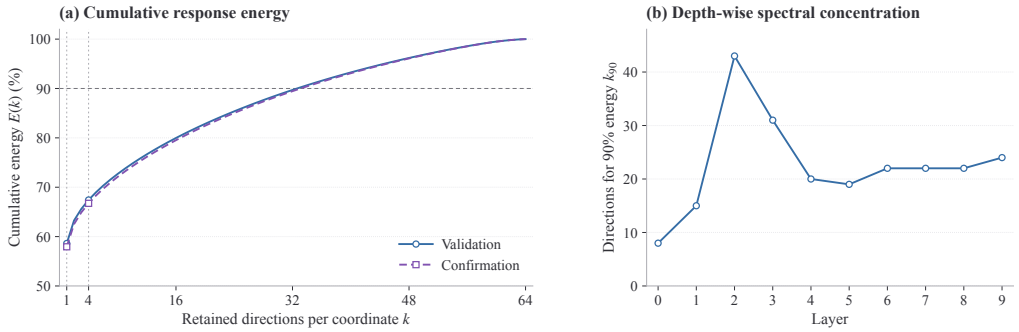


Figure 10: **Coefficient-energy spectra of the full quadratic reference.** a) Cumulative energy $E(k)$ on validation and confirmation, retaining k spectral directions per original memory coordinate. Markers highlight $k = 1$ and $k = 4$; the horizontal dashed line marks 90% energy. b) Layer-wise k_{90} on validation, using the same cumulative-energy definition within each layer.

Results. Figure 10(a) shows a response spectrum with a strong leading component and a broad tail. On validation, the leading direction captures 58.6% of the coefficient energy, while the first four directions capture 67.4%; reaching 90% requires 33 directions. The confirmation split shows nearly the same profile, with 66.7% energy captured at $k = 4$ and $k_{90} = 34$. Thus, a small number of directions captures a substantial fraction of the response energy, while additional directions continue to represent a considerable portion of the learned response structure.

Figure 10(b) further shows that this structure varies across depth: layer-wise k_{90} ranges from 8 to 43, with the broadest spectrum appearing in the early-middle layers. Together, these results characterize the full response as having a concentrated low-dimensional head together with substantial higher-dimensional structure. This provides a spectral view of the role of response modes: a small r captures the dominant response directions, whereas increasing r gives the memory access to a broader set of Query-dependent responses.

G Implementation and Pseudocode

This section summarizes the implementation of FVLA used in our experiments. We use the Value-by-Address convention of the main text; the implementation stores the fast state in the transposed

layout for kernel efficiency. Query and Key vectors below are the outputs of the host projection, causal convolution, SiLU activation, and L2 normalization. The Value vector is not L2-normalized.

G.1 Recurrent FVLA Update

For each layer and head, let $\boldsymbol{P}_1, \ldots, \boldsymbol{P}_r$ denote the response matrices and let s be the shared learnable log-scale. The implementation constructs

$$\boldsymbol{\psi}_m(\boldsymbol{u}) = \frac{\exp(s)}{\sqrt{r}}\, \boldsymbol{u} \odot (\boldsymbol{P}_m \boldsymbol{u})$$

and concatenates the linear and nonlinear coordinates into a single write and read address. Algorithm 1 gives the resulting recurrent update.

Algorithm 1 Single-token FVLA update

Require: normalized $\boldsymbol{q}_t, \boldsymbol{k}_t$, Value $\boldsymbol{v}_t$, previous state $\boldsymbol{H}_{t-1}$, response matrices $\{\boldsymbol{P}_m\}_{m=1}^{r}$, log-scale s, readout coefficient α, host log-retention g_t (GDN) or $\boldsymbol{g}_t$ (KDA), write gate β_t

for $m = 1, \ldots, r$ **do**
  $\boldsymbol{z}_m^q \leftarrow \exp(s)\, \boldsymbol{q}_t \odot (\boldsymbol{P}_m \boldsymbol{q}_t)/\sqrt{r}$
  $\boldsymbol{z}_m^k \leftarrow \exp(s)\, \boldsymbol{k}_t \odot (\boldsymbol{P}_m \boldsymbol{k}_t)/\sqrt{r}$
end for
$\boldsymbol{x}_t \leftarrow [\, \boldsymbol{k}_t; \boldsymbol{z}_1^k; \ldots; \boldsymbol{z}_r^k \,]$
$\boldsymbol{y}_t \leftarrow [\, \boldsymbol{q}_t; \alpha \boldsymbol{z}_1^q; \ldots; \alpha \boldsymbol{z}_r^q \,]$
$N_t \leftarrow \max\bigl(1, \|\boldsymbol{x}_t\|_2^2\bigr)$
$\eta_t \leftarrow \beta_t / N_t$
if GDN host **then**
  $\bar{\boldsymbol{H}}_t \leftarrow \exp(g_t) \boldsymbol{H}_{t-1}$
else if KDA host **then**
  $\boldsymbol{g}_t^F \leftarrow [\, \boldsymbol{g}_t; \ldots; \boldsymbol{g}_t \,]$ ▷ $r+1$ copies
  $\bar{\boldsymbol{H}}_t \leftarrow \boldsymbol{H}_{t-1} \operatorname{Diag}(\exp(\boldsymbol{g}_t^F))$
end if
$\boldsymbol{e}_t \leftarrow \boldsymbol{v}_t - \bar{\boldsymbol{H}}_t \boldsymbol{x}_t$
$\boldsymbol{H}_t \leftarrow \bar{\boldsymbol{H}}_t + \eta_t \boldsymbol{e}_t \boldsymbol{x}_t^\top$
$\boldsymbol{o}_t \leftarrow d^{-1/2} \boldsymbol{H}_t \boldsymbol{y}_t$
return $\boldsymbol{o}_t, \boldsymbol{H}_t$

G.2 Chunkwise Implementation

FVLA does not require a separate chunkwise recurrence. For each chunk, we first construct the expanded write and read addresses $\mathbf{X} = [\mathbf{K}; \Psi(\mathbf{K})]$ and $\mathbf{Y} = [\mathbf{Q}; \Psi_\alpha(\mathbf{Q})]$ token-wise, together with the normalized write coefficients $\eta_t = \beta_t / \max(1, \|\mathbf{x}_t\|_2^2)$. The resulting tensors are passed to the same chunkwise Gated Delta operator, with address width enlarged from d to $D = (r+1)d$ and recurrent state $\mathbf{S}$ replaced by $\mathbf{H}$. The final joint state of one chunk is carried unchanged as the initial state of the next chunk.