LEARNING TO ROUTE RECURRENT STATE FROM INPUT HISTORY

Anonymous authors
Paper under double-blind review

ABSTRACT

Can input history improve recurrent computation by controlling how stored state coordinates interact? We introduce *Input-Conditioned Graph Routing* (ICGR), a recurrent cell whose compact controller summarizes the input prefix and selects a graph-supported skew-symmetric operator from shared learned factors. The routed state current and encoded input form a bounded drive, combined with a contextual correction under an exponential retention update. Saturation and convex retention guarantee bounded main-state activations for arbitrary sequence length, without guaranteeing gradient stability. We study the mechanism through paired controls and routing interventions. Under identical parameter instantiation—not necessarily equal effective capacity—history conditioning outperforms learned time schedules and current-input-only conditioning on permuted sequential MNIST and character-level Penn Treebank. A symmetric routing control with matched support and prior spectral norm performs worse under the tested pSMNIST protocol. Donor trajectories and training-derived class templates redirect predictions toward the supplied class, while a frozen linear probe decodes class labels from the complete routing trajectory near complete-model accuracy. ICGR is competitive across four benchmarks under our evaluated configurations, although several pSMNIST baseline reruns fall below their published results and Penn Treebank shows mixed parameter efficiency. These findings identify class-bearing, history-conditioned routing signals as a task-relevant information pathway in the trained models, without establishing improved retention of target input history.

1 INTRODUCTION

Recurrent networks are natural streaming models, but evidence needed after hundreds of nonlinear transitions can contract or expand repeatedly (Pascanu et al., 2013). Structured, orthogonal, and continuous-time recurrences address propagation and timescale control (Arjovsky et al., 2016; Chang et al., 2019; Rusch & Mishra, 2021a; Erichson et al., 2021), while multiplicative and hypernetwork models make transitions input dependent (Sutskever et al., 2011; Ha et al., 2017; Costacurta et al., 2024).

We ask whether input history can route state along a fixed sparse support while preserving a structured operator and explicit retention. Input-dependent writes and decay can determine what enters the state and how long it remains, without separately controlling how stored coordinates interact. We therefore test a distinct design hypothesis: accumulated context can select state-to-state mixing, while an explicit rate path controls retention. This separation is not automatically useful—the controller could collapse to a learned clock or merely echo the current input—so the source of the routing signal and the structure of its operator require direct controls.

Input-Conditioned Graph Routing (ICGR) uses a compact controller to modulate a graph-supported skew prior in factor space, without generating a learned dense recurrent matrix. The routed state current and encoded input form a bounded drive, followed by a bounded correction and exponential retention update.

We evaluate ICGR in a common framework with fixed data handling, validation-based selection, metrics, seeds, and declared budgets. ICGR is competitive across permuted sMNIST, IMDb, noise-padded CIFAR-10, and character-level Penn Treebank under our evaluated configurations; several pSMNIST baseline reruns remain below their published results, so cross-model rankings are protocol

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.

specific. Our mechanism evidence instead centers on within-architecture tests: routing-only source controls separate input history from a learned time schedule and the current input; a support- and prior-spectral-norm-matched symmetric control tests the skew constraint; and probes, donor replay, and training-derived class templates test what the routing signal carries and whether replacing it controls predictions.

Our contributions are: (i) a recurrent cell selecting graph-supported skew operators from a learned K-factor family; (ii) exact structural and local-sensitivity characterizations plus a global forward-activation bound; and (iii) source and operator controls, routing probes and interventions, and five-seed evaluations on four tasks.

2 Related Work

Structured recurrent dynamics. Gated RNNs control writes and retention, while identity, unitary, antisymmetric, Lipschitz, and oscillatory recurrences constrain repeated state propagation (Hochreiter & Schmidhuber, 1997; Cho et al., 2014; Le et al., 2015; Arjovsky et al., 2016; Chang et al., 2019; Erichson et al., 2021; Rusch & Mishra, 2021a;b). Continuous-time models add multiscale decay or input-dependent time constants, and QORNN and HadamRNN target efficient approximately orthogonal recurrence (Mozer et al., 2017; Hasani et al., 2021; 2022; Foucault et al., 2024; 2025). ICGR inherits a skew-symmetric routed current, but does not exponentiate that current into an orthogonal transition: it enters a saturated target followed by coordinate-wise retention. The structural statement therefore concerns the instantaneous routed operator, whereas bounded activations arise from saturation and convex retention; neither statement alone guarantees gradient stability of the complete discrete update.

History-dependent weight modulation. Multiplicative and bilinear RNNs make transitions input dependent, and HyperNetworks use an auxiliary network to generate main-network weights (Sutskever et al., 2011; Wu et al., 2016; Ebrahimi & Memisevic, 2025; Ha et al., 2017). NM-RNN is especially close: a neuromodulatory subnetwork emits a low-dimensional signal that scales rank-one components of a low-rank recurrent matrix (Costacurta et al., 2024). Thus, neither a controller nor dynamic modulation is distinctive to ICGR; the difference is the modulated family and information path. ICGR scales diagonal congruences of one fixed sparse skew prior, giving a low-dimensional family with fixed support and skew structure whose members need not be low rank. Its controller is not concatenated with main-cell features: it acts through $C(z_k)h_k$ and the shared target and retention networks.

Selective state-space models. S4-family models use structured or diagonal state-space dynamics, while LRU emphasizes stable complex-diagonal recurrence (Gu et al., 2022b;a; Smith et al., 2023; Orvieto et al., 2023). Mamba makes state-space parameters functions of the input and uses a hardware-aware selective scan, so input-dependent selection is established rather than a missing capability of prior sequence models (Gu & Dao, 2024). RWKV and xLSTM likewise revisit recurrent computation through attention-like or exponential-gated memories (Peng et al., 2023; Beck et al., 2024). ICGR studies a different parameterization: a separate recurrent controller makes z_k an explicit function of the observed prefix and modulates non-diagonal mixing among hidden coordinates inside a nonlinear cell. Our controls test this particular history-to-mixing path; they do not establish a general limitation of selective state-space models.

Graph-structured recurrence. Gated graph recurrent networks and Graph Neural ODEs propagate features over graphs that structure the modeled input domain (Ruiz et al., 2020; Poli et al., 2019). In ICGR, the fixed ER-like graph instead indexes coordinates of one hidden state: it is an internal routing prior, not an observed relational graph. History changes supported edge weights through shared factors but neither learns the support nor gives nodes application-level semantics. This limits graph-specific interpretation while isolating structured hidden-state routing.

3 Input-Conditioned Graph-Routed Recurrence

Let h_k denote the main state before x_k, with h_1 initial and h_{T+1} following x_T. ICGR also contains a controller state $c_k \in \mathbb{R}^{D_c}$. At each step, its GRU summarizes the input history, emits K positive

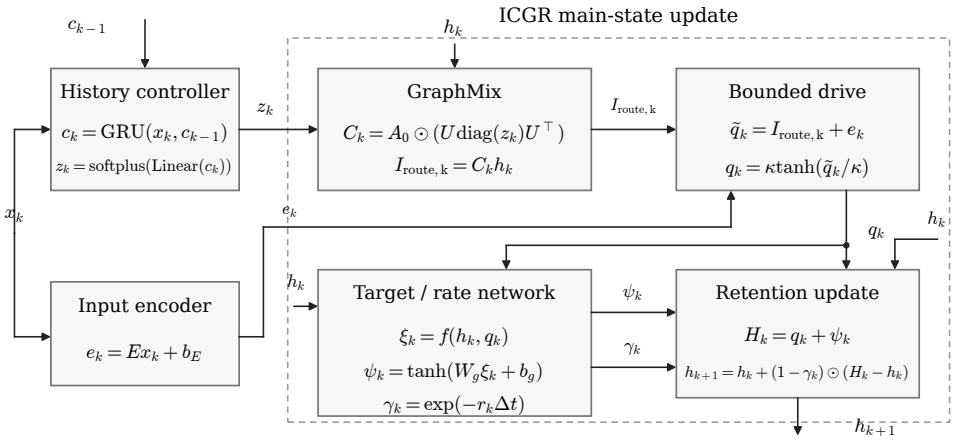

Figure 1: Input-conditioned graph routing in ICGR. The controller summarizes the prefix and emits z_k, selecting a graph-supported skew operator in GraphMix. The routed current and encoded input form q_k, which enters the target directly and the shared target/rate network. Controller information reaches the main state only through routed current; f denotes Equations (7)–(8).

modulation coordinates, and conditions the main-state update:

$$c_k = \mathrm{GRUCell}(x_k, c_{k-1}), \qquad z_k = \mathrm{softplus}(W_{\mathrm{ctrl}} c_k + b_{\mathrm{ctrl}}), \qquad h_{k+1} = F(h_k, x_k, z_k). \quad (1)$$

Here c_k summarizes $x_{1:k}$ and z_k selects a structured transition shared across the N hidden units. The reported models use $N = 128$, $K = 16$, $D_c = 64$, $\Delta t = 0.01$, and $\kappa = 10$; task-specific widths and initialization are in Appendix A.1.

3.1 Graph-supported skew routing

We sample a fixed directed ER-like graph prior and skew-symmetrize it to obtain A_0; graph sampling and empty-row repair are detailed in Appendix A.1. We learn $U \in \mathbb{R}^{N \times K}$. For example ℓ at step k, define

$$M_k^{(\ell)} = U \,\mathrm{diag}(z_k^{(\ell)}) U^\top. \quad (2)$$

The routed operator and current are

$$C_k^{(\ell)} = A_0 \odot M_k^{(\ell)}, \qquad I_{\mathrm{route},k}^{(\ell)} = C_k^{(\ell)} h_k^{(\ell)}. \quad (3)$$

They can be evaluated without materializing $M_k^{(\ell)}$:

$$I_{\mathrm{route},k}^{(\ell)} = \sum_{r=1}^{K} z_{k,r}^{(\ell)} U_{:,r} \odot \left[A_0 \left(h_k^{(\ell)} \odot U_{:,r} \right) \right]. \quad (4)$$

Equations (3)–(4) form one `GraphMix` module. The prior A_0 fixes the support and a signed coefficient on each supported edge, while the learned factor can change the final signs and magnitudes. Thus history changes weights within a fixed support while preserving skew-symmetry. We suppress the per-example superscript below.

Expressivity–cost tradeoff. Equation (4) restricts routing to an at-most-K-dimensional span of diagonal congruences of A_0; positive $z_{k,r}$ further restricts the reachable subset. It cannot create edges outside A_0's support, though cancellations can remove them, and uses NK learned factor parameters.

The encoded input is added to the routed current before saturation:

$$\tilde{q}_k = I_{\mathrm{route},k} + Ex_k + b_E, \quad (5)$$

where $E \in \mathbb{R}^{N \times d_x}$. This is the complete graph-dependent path in canonical ICGR, which has no Laplacian branch or direct controller-feature shortcut. The variants in the matched mechanism ablation are defined separately in Section 6.

3.2 TARGET FORMATION AND EXPONENTIAL RETENTION

Before entering the target and rate networks, the composite drive is smoothly bounded coordinate-wise:

$$q_k = \kappa \tanh(\widetilde{q}_k/\kappa). \tag{6}$$

Here $\widetilde{q}_k$ is the pre-saturation drive and q_k is the drive used by the recurrent update. All reported ICGR models use this saturation, and controller information enters the main cell only through q_k. The main cell projects h_k and q_k into a common width D and applies

$$v_k = W_h h_k + b_h + W_q q_k + b_q, \tag{7}$$

$$\begin{aligned} y_k &= W_1 v_k + b_1, \\ \xi_k &= W_2 \,\mathrm{SiLU}(\mathrm{LayerNorm}(y_k)) + b_2, \end{aligned} \tag{8}$$

$$\begin{aligned} r_k &= \mathrm{softplus}(W_r \xi_k + b_r) + \epsilon, \\ \psi_k &= \tanh(W_g \xi_k + b_g), \end{aligned} \tag{9}$$

and forms the target from an additive routed-drive path and a bounded correction:

$$\mathcal{H}_k = q_k + \psi_k. \tag{10}$$

Layer dimensions are recorded in Appendix A.1. The direct q_k term preserves the bounded routed current outside the final $\tanh$ correction, while both the correction and retention rate remain context dependent through the routed drive. The rate head directly represents the positive effective rate r_k. The retention and exact implemented update are

$$\gamma_{k,i} = \exp(-r_{k,i}\Delta t)\,, \qquad h_{k+1} = h_k + (1-\gamma_k) \odot (\mathcal{H}_k - h_k). \tag{11}$$

The rate bias sets a nominal initial half-life of 100 updates (log-spaced 1–1000 on noise-CIFAR), while rates remain trainable; Appendix A.1 gives the formula and initialization details.

Computational characteristics. Excluding the external input encoder E, task embedding, and classifier, ICGR has $O(NK + ND + D^2 + D_c^2 + d_x D_c + K D_c)$ recurrent parameters. It creates no per-example $N \times N$ operator, but the dense adjacency contraction costs $O(mN^2K)$ per step and $O(N^2)$ graph-buffer memory. The measured implementation is slower than fused LSTM and parallel S4D kernels; full profiling is in Appendix A.2.

4 STRUCTURE AND SENSITIVITY OF HISTORY-CONDITIONED ROUTING

Proposition 1. *Let $p_r = U_{:,r}$ and $D_{p_r} = \mathrm{diag}(p_r)$. The modulated operator defined in Equation (3) satisfies*

$$C_k^{(\ell)} = A_0 \odot \left(U \,\mathrm{diag}(z_k^{(\ell)}) U^\top\right) = \sum_{r=1}^{K} z_{k,r}^{(\ell)} D_{p_r} A_0 D_{p_r}. \tag{12}$$

Consequently, $C_k^{(\ell)}$ is skew-symmetric, its support is contained in that of A_0, and $h^\top C_k^{(\ell)} h = 0$.

Equation (12) gives an at-most-K-dimensional factorized span; positive z_k further restricts the reachable subset. This is a property of the routed current, not a stability guarantee for the discrete update. The coordinate-wise-gate commutator and local Jacobian analysis are in Appendix B.

Modulation dimension is not matrix rank. Let $B_r = D_{p_r} A_0 D_{p_r}$, so $C(z) = \sum_{r=1}^{K} z_r B_r$. The family has at most K degrees of linear modulation, but its members need not have rank at most K. For example, when $K = 1$, every entry of $u = p_1$ is nonzero, and $z_1 > 0$, the diagonal matrix D_u is invertible and

$$\mathrm{rank}\, C(z) = \mathrm{rank}(z_1 D_u A_0 D_u) = \mathrm{rank}\, A_0. \tag{13}$$

Thus a low-dimensional controller selects from a low-dimensional operator family whose members can still act across a high-dimensional state. This is a parameterization property, not a memory-capacity or performance guarantee.

Proposition 2. *Fix the current main state h and input x, and regard z as an intervenable modulation variable. Define*

$$V(h) = [B_1 h \quad \cdots \quad B_K h] \in \mathbb{R}^{N \times K}, \qquad C(z)h = V(h)z. \tag{14}$$

Write $\widetilde{q} = V(h)z + Ex + b_E$, $q = \kappa \tanh(\widetilde{q}/\kappa)$, $\eta = 1 - \gamma$, $\mathcal{H} = q + \psi$, and $D_v = \operatorname{diag}(v)$. Holding h fixed, let

$$L = \frac{\partial F}{\partial q} = D_\eta + D_\eta J_{\psi,q} + D_{\mathcal{H}-h} J_{\eta,q}, \qquad S = \operatorname{diag}\left[\operatorname{sech}^2(\widetilde{q}/\kappa)\right]. \tag{15}$$

Then the complete implemented cell, including the correction and adaptive retention rate, satisfies

$$\boxed{\frac{\partial F(h,x,z)}{\partial z} = LSV(h).} \tag{16}$$

For $h \neq 0$,

$$\operatorname{rank} \frac{\partial F}{\partial z} \leq \operatorname{rank} V(h) \leq \min\{K, N-1\}. \tag{17}$$

Moreover, any two legal signals with $V(h)(z - z') = 0$ produce exactly the same next state at fixed (h, x); when $h = 0$, the current update is independent of z.

Each raw routing direction $B_r h$ is orthogonal to h because B_r is skew-symmetric. The final perturbation need not remain orthogonal after the saturation and cell Jacobian LS. Proposition 2 therefore bounds local action dimension, not information capacity: history changes the update through state-dependent directions $B_r h$, rather than through an additive controller feature. The proof is in Appendix B.2.

Corollary 4.1. *Along a fixed input and routing trajectory, let $J_k = \partial F_k / \partial h_k$ and $G_k = \partial F_k / \partial z_k$. With zero initial-state perturbation, the first variation induced by routing perturbations is*

$$\delta h_{T+1} = \sum_{t=1}^{T} J_T J_{T-1} \cdots J_{t+1} G_t \, \delta z_t, \tag{18}$$

where the product is the identity for $t = T$.

Writing $\Phi_{T,t} = J_T \cdots J_{t+1}$ and concatenating the propagated directions gives

$$\delta h_{T+1} = \underbrace{[\Phi_{T,1} G_1 \quad \cdots \quad \Phi_{T,T} G_T]}_{\mathcal{R}_T} \begin{bmatrix} \delta z_1 \\ \vdots \\ \delta z_T \end{bmatrix}, \qquad \operatorname{rank} \mathcal{R}_T \leq \min\{N, TK\}. \tag{19}$$

Each G_t has rank at most K, but propagation can orient different time steps along different state directions, so the trajectory-level action need not be K-dimensional. This is neither a controllability nor a memory-capacity claim. Equations (18)–(19) remain local first-order descriptions: finite donor replacement is not governed exactly by them, and they do not imply that early segments must dominate. The full derivation is in Appendix B.2.

Proposition 3. *Let $\eta_k = 1 - \gamma_k$. Suppose the implemented target satisfies*

$$\mathcal{H}_k = q_k + \psi_k, \qquad q_k = \kappa \tanh(\widetilde{q}_k/\kappa), \qquad \psi_k = \tanh(\cdot), \tag{20}$$

and $0 \leq \eta_{k,i} \leq 1$. Then, for every input sequence, controller trajectory, and $k \geq 1$,

$$\|h_k\|_\infty \leq \max\{\|h_1\|_\infty, \kappa + 1\}. \tag{21}$$

The bound follows by induction from saturation, the bounded correction, and coordinate-wise convex retention, not skew-symmetry; it implies neither convergence nor gradient stability. The proof and complete local analysis are in Appendix B.

5 Experiments

We test complementary aspects of recurrent computation: a dense permuted image stream, image retention through 968 distractor steps, sentiment classification, and next-character prediction. A common framework fixes data construction and splits, metrics, seed protocol, validation-only checkpoint selection, and maximum training budgets. Architectures may use limited validation-selected optimizer adjustments informed by published settings; the result is a best-effort accuracy comparison, not an equal-compute comparison. Appendix A records the final configurations.

Reporting scope. Main comparisons and the six-variant ablation in Table 9 aggregate five runs. Conditioning controls use three paired seeds; replay uses those checkpoints and averages donor maps within checkpoint. Tables state their replication units. Hyperparameters and checkpoints use training and validation data only; each selected checkpoint is evaluated once on test data before the declared interventions.

5.1 PERMUTED 784-STEP IMAGE STREAMS

Permuted sMNIST applies one fixed pixel permutation before recurrence. Table 1 compares recurrent and state-space baselines at the widths and budgets stated in its caption.

We use one permutation (seed 923), split seed 42, training seeds 0–4, and leave the official test split untouched.

Table 1: Results under our evaluation protocol on permuted sequential MNIST (mean ± standard deviation over five runs). All models use width 128 except HadamRNN (512); coRNN and HadamRNN use 100 epochs and the remaining reruns use 50.

Model	Params	Test acc. (%)
Antisym. RNN	17.9k	94.63 ± 0.34
Lipschitz RNN	51.0k	94.81 ± 0.29
coRNN	34.3k	93.77 ± 0.36
UniCORNN (3 layers)	35.3k	95.32 ± 0.45
LSTM	85.1k	90.63 ± 0.44
S4D (complex)	117.8k	94.46 ± 0.47
HyperRNN	39.6k	91.18 ± 0.31
QORNN	17.8k	93.02 ± 0.35
HadamRNN (512)	6.7k	93.56 ± 0.48
ICGR	59.1k	$\mathbf{96.32 \pm 0.24}$

Several rerun baselines fall below their published pSMNIST results; Appendix A.8 provides a source-versus-rerun comparison and documents protocol differences. The rankings in Table 1 therefore apply to our evaluated configurations and training protocols, rather than the best reported performance of each architecture. Our evidence for history-conditioned routing primarily comes from the paired within-architecture controls in Section 6.

5.2 NOISE-PADDED CIFAR-10

Noise-padded CIFAR-10 tests whether a recurrent model can retain visual evidence across a long distractor interval (Chang et al., 2019; Rusch & Mishra, 2021a). Each CIFAR-10 image (Krizhevsky & Hinton, 2009) is fed row by row: the 32 pixels and three color channels in each row are flattened into a 96-dimensional input, producing 32 image steps. We then append 968 noise vectors to obtain a sequence of length 1000, and classify from the final recurrent state.

Following the released coRNN benchmark, every model receives the same Gaussian suffix (seed 0), shared across examples and splits. We use split seed 42, leave the official test split unchanged, and use graph seed 0 for ICGR headline runs.

Table 2 reports five-seed results under the selection policy above; Appendix A records the model-specific optimization details.

ICGR obtains the highest mean accuracy, 2.06 percentage points above the width-256 Lipschitz RNN and 2.11 points above the width-256 coRNN.

5.3 NATURAL-LANGUAGE SEQUENCE MODELING

IMDb (Maas et al., 2011) uses the supplied train/test split, an $80/20$ train/validation split, length-500 sequences, and a shared trainable 128-dimensional embedding; all recurrent widths are 128. The comparison includes IndRNN (Li et al., 2018). Because the embedding dominates, Table 3(a) omits parameter counts and makes no capacity-matched claim. Appendix A.3 records only ICGR's exact non-embedding count, not baseline-wide matching.

Table 2: Noise-padded CIFAR-10 at sequence length 1000 (mean $\pm$ standard deviation, $n = 5$). All checkpoints are selected on validation accuracy.

Model	Configuration	Active params	Test acc. (%)
Antisym. RNN	$H = 256$	159.0k	50.16 ± 0.31
Lipschitz RNN	$H = 128$	46.5k	57.35 ± 0.38
Lipschitz RNN	$H = 256$	158.5k	58.97 ± 0.29
coRNN	$H = 128$	46.5k	56.42 ± 0.34
coRNN	$H = 256$	158.5k	58.92 ± 0.31
HadamRNN	$H = 1024$	110.6k	39.21 ± 0.52
NM-RNN	$H = 128$	29.0k	47.54 ± 0.25
NM-RNN	$H = 256$	46.6k	50.65 ± 0.32
QORNN	$H = 128$	30.0k	54.51 ± 0.35
QORNN	$H = 256$	92.7k	56.81 ± 0.44
UniCORNN	$H = 128$	47.5k	58.08 ± 0.29
UniCORNN	$H = 256$	160.5k	57.83 ± 0.33
ICGR	$H = 128$	89.5k	$\mathbf{61.03 \pm 0.38}$

Character-level Penn Treebank (PTB) (Marcus et al., 1993; Foucault et al., 2024) uses independent length-150 sentences over 50 characters, resetting state between lines and masking padded BPC targets. ICGR uses backbone width $D = 128$. The main comparison uses batch size 256 and the source controls use 128. Because BPC is capacity-sensitive, Table 3 reports active parameters and several widths; preprocessing, BPC, and optimization details are in Appendix A.11.

Table 3: Natural-language tasks. (a) IMDb test accuracy. (b) Sentence-level character Penn Treebank test BPC (lower is better). All entries are means over five runs.

(a) IMDb ($n = 5$)

Model	Configuration	Test acc. (%)
CfC	$H = 128$	85.14 ± 0.54
coRNN	$H = 128$	81.32 ± 0.17
IndRNN	$H = 128$	86.93 ± 0.35
LSTM	$H = 128$	85.04 ± 0.99
LTC	$H = 128$	83.41 ± 0.38
HyperRNN	$H = 128$	80.48 ± 0.11
NM-RNN	$H = 128$	87.15 ± 0.71
QORNN	$H = 128$	85.88 ± 0.12
HadamRNN	$H = 128$	87.02 ± 0.56
ICGR	$H = 128$	$\mathbf{88.06 \pm 0.21}$

(b) Character PTB ($n = 5$)

Model	Configuration	Active params	Test BPC ↓
HadamRNN	$H = 1024$	104.5k	3.243 ± 0.003
LSTM	$H = 128$	98.6k	1.692 ± 0.002
LSTM	$H = 256$	328.2k	1.563 ± 0.002
GRU	$H = 128$	75.6k	1.660 ± 0.001
GRU	$H = 256$	249.4k	$\mathbf{1.483 \pm 0.002}$
Mamba	$H = 128$	130.0k	1.689 ± 0.002
Mamba	$H = 256$	356.4k	1.544 ± 0.001
QORNN	$H = 128$	29.2k	1.964 ± 0.002
QORNN	$H = 256$	91.2k	1.701 ± 0.002
S4D	$H = 128$	112.7k	1.731 ± 0.002
S4D	$H = 256$	356.4k	1.559 ± 0.003
NM-RNN	$H = 256$	58.7k	1.844 ± 0.002
ICGR	$H = 128$	137.7k	1.641 ± 0.001

ICGR has the highest IMDb mean, with NM-RNN and HadamRNN respectively 0.91 and 1.04 percentage points lower. On PTB, the width-256 GRU obtains the lowest BPC; ICGR at width 128 improves over its width-matched GRU but does not outperform the larger conventional gated baseline. The spread across widths confirms that BPC is strongly capacity-sensitive.

6 Mechanism Ablations

We begin with paired controls in the canonical routing-only, Laplacian-free cell. The earlier six-variant dual-path results remain supporting comparisons in Appendix A.4, because their rows alter more than one information path.

Conditioning source. The first control asks whether the gain requires accumulated input history, rather than a learned time schedule or the current observation alone. History uses the observed prefix; time-only feeds zeros to the same recurrent controller, preserving its recurrence as a position-dependent schedule; and current-input resets the controller state at every position. All three instantiate identical architecture and parameters and share the graph, initialization, minibatch order, and optimization within each paired seed. Only controller information changes; this can alter how input or recurrent weights participate effectively, so it is not proof of equal effective capacity. In Table 4(a), history wins every paired seed on both tasks. Relative to time-only it gains 2.427 ± 0.127 pSMNIST

points and reduces PTB BPC by 0.107 ± 0.009; relative to current-input the corresponding differences are 2.027 ± 0.309 points and 0.036 ± 0.002 BPC.

Operator structure. The second control asks whether the skew constraint matters under otherwise paired routing. Starting from each skew prior, we retain its signed strict upper triangle, mirror that triangle onto the lower half without negation to obtain a symmetric matrix, and rescale the result to match the skew prior's spectral norm. This preserves the graph draw and nonzero support while changing skew-symmetry to symmetry; learned parameters, their initialization, minibatches, and optimization remain paired. Table 4(b) shows an improvement for skew in all three paired seeds, averaging 9.43 ± 0.53 pSMNIST points. The comparison is specific to this construction and protocol: matching A_0 does not force the modulated C_k or routed-current scales to match during training.

Table 4: Paired mechanism controls ($n = 3$ training seeds). (a) Controller information in the canonical routing-only cell. (b) Support- and prior-spectral-norm-matched operator structure on pSMNIST.

(a) Conditioning source

Controller source	pSMNIST acc. (%) ↑	PTB BPC ↓
History	$\mathbf{96.30 \pm 0.11}$	$\mathbf{1.640 \pm 0.007}$
Time only	93.87 ± 0.23	1.747 ± 0.011
Current input only	94.27 ± 0.21	1.676 ± 0.005

(b) Operator structure

Routing prior	pSMNIST acc. (%)
Skew	$\mathbf{96.30 \pm 0.11}$
Symmetric	86.87 ± 0.63

Routing replay and class templates. We intervene on three trained pSMNIST checkpoints without retraining, keeping each target's input, initial state, and readout fixed while replacing its complete routing trajectory $z_{1:T}$. Ten deterministic same- or different-class donor maps are averaged within checkpoint; the independently trained checkpoints are the replicates. Self-replay checks the implementation. To remove donor-instance detail, we also average training-set routing in log space to estimate one template per class. Wrong-class template replay supplies the template for a designated different class; class-shift adds the designated-class mean and subtracts the target-class mean from the target trajectory in log space. Test labels construct these diagnostics but are not model inputs; Appendix A.6 gives full definitions.

Table 5: Routing interventions on pSMNIST (mean ± standard deviation across three checkpoints; maps averaged within checkpoint; templates estimated from training data only).

Routing intervention	Target acc. (%)	Designated-class acc. (%)
Normal	96.300 ± 0.111	—
Different-class donor trajectory	0.757 ± 0.098	95.508 ± 0.286
Wrong-class template	0.488 ± 0.572	96.098 ± 4.302
Class-shift	11.761 ± 0.567	58.216 ± 0.578

Self-replay reproduces every checkpoint's logits, and same-class replay preserves aggregate accuracy (Appendix A.5). Different-class trajectories and templates instead redirect predictions to the designated class. The wrong-class template is decisive: a class-shared pattern reaches 96.10% designated-class accuracy without a donor instance's trajectory. This effect enters through routed state mixing; the absence of a direct controller-feature branch does not prevent the multiplicative routing channel from carrying class information. Class-shift supports transfer of a constructed class-associated component, not a unique decomposition (Appendix A.6).

Segmented routing replay. At equal 75% duration, donor prefixes yield $85.84 \pm 1.08\%$ designated-class accuracy versus $64.46 \pm 2.59\%$ for donor suffixes. This persistent early-segment influence is observed, not implied by Corollary 4.1, and does not distinguish memory preservation from early state bias; complete results are in Table 12.

Frozen routing probes. A validation-selected linear probe on the complete flattened routing trajectory reaches $95.253 \pm 0.107\%$, versus $96.300 \pm 0.111\%$ for ICGR and $43.893 \pm 0.735\%$ for the

final-c_T probe. Under this protocol, the c_T probe does not reproduce complete-model performance and understates class information decodable from $z_{1:T}$. This establishes decodability, not use of the same decoder or improved retention of target history (Appendix A.7).

Routed-branch removal. Setting $A_0 = 0$ removes the controller's only path into the canonical main cell and drops accuracy by 8.71 points. Graph-seed sensitivity is in Appendix A.7.

7 Limitations and Open Questions

ICGR's positive, at-most-K-dimensional modulation and fixed ER-like support limit expressivity. The symmetric control matches the prior, not stepwise operator or current scales, and covers one task and protocol. Benchmarks permit architecture-specific optimization; noise-CIFAR uses one distractor process and sentence-level PTB is not comparable to continuous-stream BPTT. Profiling is implementation specific. Appendix A records protocols and extended controls.

Replay uses constructed, potentially off-distribution input–routing pairs. It shows that replacing the routing channel with donor trajectories or class-shared templates can control predictions in trained pSMNIST models. This intervention-level sufficiency does not imply that natural inference relies on one unique class code, that the complete model uses the trajectory probe's decoder, or that routing improves retention of target history. Activation boundedness comes from saturation and convex retention, not skew-symmetry, and implies neither convergence nor gradient stability. Broader supports, capacity-matched language comparisons, and sparse or fused kernels remain open.

8 Conclusion

ICGR uses input history to select graph-supported skew routing under bounded drive and retention. A local factorization identifies how modulation enters the update through state-dependent directions. Paired controls favor history conditioning and skew routing over tested alternatives; probes, donor replay, and class-template interventions show that the routing signal carries class information and can control predictions. These findings establish a task-relevant information pathway, not improved retention of target history. Broader graph families, language scaling, and efficient sparse kernels remain open.

AI USE STATEMENT

We used generative AI tools to assist with drafting and reviewing experimental code, clarifying mathematical notation and proof exposition, organizing supplementary material, and editing the manuscript for clarity and LaTeX formatting. AI tools also provided feedback on the description of experimental methods and the interpretation of results. The authors reviewed the AI-assisted material and take responsibility for the final manuscript, proofs, code, and reported results.

REPRODUCIBILITY STATEMENT

The main text and Appendix A specify the implemented update, final configurations, preprocessing, evaluation metrics, and validation-based model-selection protocol. The anonymous supplementary archive provides the model and training code, task entry points, dependency list, explicit training, graph, data, and split seeds, the Penn Treebank preprocessing and padding masks, ablation and diagnostic implementations, unit tests, and performance-measurement scripts. In particular, the clean pSMNIST controller-source and operator-structure controls are implemented by `conditioning_source_ablation/run_psmnist.py` and `conditioning_source_ablation/run_psmnist_operator_structure.py`, using `ICGRClassifier` from `conditioning_source_ablation/model.py`. Datasets, pretrained checkpoints, and complete per-epoch logs are not included. The supplied scripts support rerunning training and evaluation under the documented protocols.

REFERENCES

Martin Arjovsky, Amar Shah, and Yoshua Bengio. Unitary evolution recurrent neural networks. In *Proceedings of the International Conference on Machine Learning*, 2016.

Maximilian Beck, Korbinian Pöppel, Markus Spanring, Andreas Auer, Oleksandra Prudnikova, Michael Kopp, Günter Klambauer, Johannes Brandstetter, and Sepp Hochreiter. xLSTM: Extended long short-term memory. In *Advances in Neural Information Processing Systems*, volume 37, 2024. doi: 10.52202/079017-3417.

Bo Chang, Minmin Chen, Eldad Haber, and Ed H. Chi. AntisymmetricRNN: A dynamical system view on recurrent neural networks. In *International Conference on Learning Representations*, 2019.

Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pp. 1724–1734, 2014. doi: 10.3115/v1/D14-1179.

Julia C. Costacurta, Shaunak Bhandarkar, David Zoltowski, and Scott W. Linderman. Structured flexibility in recurrent neural networks via neuromodulation. In *Advances in Neural Information Processing Systems*, volume 37, 2024. doi: 10.52202/079017-0062.

M. Reza Ebrahimi and Roland Memisevic. Revisiting bi-linear state transitions in recurrent neural networks. In *Advances in Neural Information Processing Systems*, volume 38, 2025.

N. Benjamin Erichson, Omri Azencot, Alejandro Queiruga, Liam Hodgkinson, and Michael W. Mahoney. Lipschitz recurrent neural networks. In *International Conference on Learning Representations*, 2021.

Armand Foucault, Franck Mamalet, and François Malgouyres. Quantized approximately orthogonal recurrent neural networks. *CoRR*, abs/2402.04012, 2024. doi: 10.48550/arXiv.2402.04012. URL `https://arxiv.org/abs/2402.04012`.

Armand Foucault, François Malgouyres, and Franck Mamalet. HadamRNN: Binary and sparse ternary orthogonal RNNs. In *The Thirteenth International Conference on Learning Representations*, 2025. URL `https://openreview.net/forum?id=amOpepqmSl`.

Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. In *First Conference on Language Modeling*, 2024. URL `https://openreview.net/forum?id=tEYskw1VY2`.

Albert Gu, Karan Goel, Ankit Gupta, and Christopher Ré. On the parameterization and initialization of diagonal state space models. In *Advances in Neural Information Processing Systems*, volume 35, pp. 35971–35983, 2022a. doi: 10.52202/068431-2607.

Albert Gu, Karan Goel, and Christopher Re. Efficiently modeling long sequences with structured state spaces. In *International Conference on Learning Representations*, 2022b.

David Ha, Andrew Dai, and Quoc V. Le. Hypernetworks. In *International Conference on Learning Representations*, 2017.

Ramin Hasani et al. Liquid time-constant networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2021.

Ramin Hasani et al. Closed-form continuous-time neural networks. *Nature Machine Intelligence*, 2022.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8): 1735–1780, 1997. doi: 10.1162/neco.1997.9.8.1735.

Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009. URL `https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf`.

Quoc V. Le, Navdeep Jaitly, and Geoffrey E. Hinton. A simple way to initialize recurrent networks of rectified linear units. *arXiv preprint arXiv:1504.00941*, 2015.

Shuai Li, Wanqing Li, Chris Cook, Ce Zhu, and Yanbo Gao. Independently recurrent neural network (IndRNN): Building a longer and deeper RNN. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 5457–5466, 2018. URL https://openaccess.thecvf.com/content_cvpr_2018/html/Li_Independently_Recurrent_Neural_CVPR_2018_paper.html.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Proceedings of the ACL-HLT*, 2011.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. Building a large annotated corpus of english: The Penn Treebank. *Computational Linguistics*, 19(2):313–330, 1993.

Michael C. Mozer, Denis Kazakov, and Robert V. Lindsey. Discrete event, continuous time RNNs. *arXiv preprint arXiv:1710.04110*, 2017.

Antonio Orvieto, Samuel L. Smith, Albert Gu, Anushan Fernando, Caglar Gulcehre, Razvan Pascanu, and Soham De. Resurrecting recurrent neural networks for long sequences. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 26670–26698. PMLR, 2023. URL https://proceedings.mlr.press/v202/orvieto23a.html.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. In *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, pp. 1310–1318, 2013. URL https://proceedings.mlr.press/v28/pascanu13.html.

Bo Peng et al. RWKV: Reinventing rnns for the transformer era. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 14048–14077. Association for Computational Linguistics, 2023. doi: 10.18653/v1/2023.findings-emnlp.936. URL https://aclanthology.org/2023.findings-emnlp.936/.

Michael Poli et al. Graph neural ordinary differential equations. *arXiv preprint arXiv:1911.07532*, 2019.

Luana Ruiz, Fernando Gama, and Alejandro Ribeiro. Gated graph recurrent neural networks. *IEEE Transactions on Signal Processing*, 68:6303–6318, 2020. doi: 10.1109/TSP.2020.3033962.

Konstantin Rusch and Siddhartha Mishra. Coupled oscillatory recurrent neural network (coRNN): An accurate and (gradient) stable architecture for learning long time dependencies. In *International Conference on Learning Representations*, 2021a.

Konstantin Rusch and Siddhartha Mishra. UnICORNN: A recurrent model for learning very long time dependencies. In *Proceedings of the International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 9168–9178. PMLR, 2021b. URL https://proceedings.mlr.press/v139/rusch21a.html.

Jimmy T. H. Smith, Andrew Warrington, and Scott W. Linderman. Simplified state space layers for sequence modeling. In *International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=Ai8Hw3AXqks.

Ilya Sutskever, James Martens, and Geoffrey E. Hinton. Generating text with recurrent neural networks. In *Proceedings of the 28th International Conference on Machine Learning*, pp. 1017–1024, 2011. URL https://icml.cc/2011/papers/524_icmlpaper.pdf.

Yuhuai Wu, Saizheng Zhang, Ying Zhang, Yoshua Bengio, and Ruslan Salakhutdinov. On multiplicative integration with recurrent neural networks. In *Advances in Neural Information Processing Systems*, volume 29, pp. 2856–2864, 2016. URL https://proceedings.neurips.cc/paper/2016/hash/f69e505b08403ad2298b9f262659929a-Abstract.html.

A Experimental Recipes

Selection policy and provenance. This appendix records the final configurations used for the reported runs, not an exhaustive search audit. Published settings are used where available; otherwise, configurations are selected by validation performance only, without consulting the test split. Every aggregate in the main benchmark tables uses seeds 0–4 and the validation-selected checkpoint from each run.

A.1 ICGR configuration and initialization

The fixed graph prior begins with a directed Bernoulli matrix A_{raw} of keep probability 0.2, sampled once from the recorded graph seed. Before skew-symmetrization, every empty row is repaired by inserting one uniformly sampled entry. This repair applies to A_{raw} and does not guarantee that $A_0 = (A_{\mathrm{raw}} - A_{\mathrm{raw}}^\top)/2$ has no empty rows: self-loops and equal reciprocal entries vanish under skew-symmetrization.

The affine-map shapes are $W_{\mathrm{ctrl}} \in \mathbb{R}^{K \times D_c}$, $E \in \mathbb{R}^{N \times d_x}$, $W_h, W_q \in \mathbb{R}^{D \times N}$, $W_1, W_2 \in \mathbb{R}^{D \times D}$, and $W_r, W_g \in \mathbb{R}^{N \times D}$, with conforming biases.

Table 6: Task-specific ICGR configuration. L is the nominal initial retention half-life.

Task	K	D_c	D	Δt	κ	L	ϵ
pSMNIST	16	64	64	0.01	10	100	10^{-3}
Noise-CIFAR-10	16	64	64	0.01	10	1–1000	10^{-3}
IMDb	16	64	64	0.01	10	100	10^{-3}
Character PTB	16	64	128	0.01	10	100	10^{-3}

The factor matrix U uses independent standard-normal entries, E uses Xavier-normal weights at gain 0.1, and remaining GRU and affine weights use PyTorch defaults. For nominal half-life L_i, the rate bias is

$$b_{r,i} = \mathrm{softplus}^{-1}\left(\frac{\log 2}{\Delta t L_i} - \epsilon\right). \tag{22}$$

When $W_r \xi_k = 0$, this gives $r_i = \log 2/(\Delta t L_i)$ and $\gamma_i^{L_i} = 1/2$. Because W_r is not zero-initialized, L_i is a bias-calibrated nominal timescale rather than an identical initial rate for every sample and channel.

A.2 Implementation profiling

Table 7 reports same-hardware measurements at the full pSMNIST length. We use synchronized CUDA events and exclude host-to-device transfer and one-time compilation. Training time includes the forward pass, cross-entropy loss, backward pass, gradient clipping, and an Adam update. Each entry is the mean over three independently launched processes after ten warmup iterations. Baseline weights are randomly initialized because the measured shape-fixed dense kernels do not use weight-dependent execution paths; ICGR uses the validation-selected checkpoint also used to verify numerical equivalence of its chunked and one-step implementations.

Table 7: Steady-state pSMNIST implementation measurements at $T = 784$ and batch size 256 on one NVIDIA RTX 5090 in FP32 (PyTorch 2.8, CUDA 12.8). Latencies are milliseconds per batch (mean $\pm$ process-level standard deviation, $n = 3$); memory is peak allocated GPU memory.

Model	Params	Train ms ↓	Infer ms ↓	Train MB ↓	Infer MB ↓
LSTM	85.1k	7.92 ± 0.12	3.10 ± 0.01	2872.4	2184.6
S4D (complex)	117.8k	15.06 ± 0.01	4.72 ± 0.00	1940.0	1195.4
QORNN	17.8k	117.98 ± 0.47	26.81 ± 1.02	117.0	18.3
Lipschitz RNN	51.0k	418.88 ± 17.06	144.17 ± 2.41	412.5	18.5
UniCORNN	35.3k	1286.78 ± 0.10	220.13 ± 2.60	1784.2	311.3
NM-RNN	10.7k	679.26 ± 1.99	107.77 ± 1.20	593.0	166.5
ICGR (chunked)	59.1k	180.06 ± 0.05	44.39 ± 0.04	1352.9	853.4

ICGR is $2.33\times$ faster to train and $3.25\times$ faster at inference than the measured Lipschitz RNN, and it is faster than the NM-RNN and UniCORNN implementations. QORNN is $1.53\times$ and $1.66\times$ faster than ICGR for training and inference; the present QORNN code uses FP32 fake quantization rather than a deployed low-bit kernel. ICGR's training peak is 1.35 GB, below the 2.87 GB LSTM and 1.94 GB S4D peaks, although memory is backend- and workspace-dependent.

Chunking changes implementation overhead, not the recurrence. At batch size 256, lengths 196, 392, and 784 give ICGR training latencies of 43.18, 86.09, and 180.06 ms and inference latencies of 11.32, 22.22, and 44.39 ms. Relative to the one-step implementation, speedups are 5.20–$5.31\times$ for training and 4.31–$4.36\times$ for inference. Scaling is approximately linear over this range; dense graph contraction and the sequential controller remain optimization targets.

A.3 Instantiated and active parameter counts

Table 8 distinguishes all instantiated trainable parameters from parameters that participate in each variant's forward graph. We define a parameter as active if backpropagation through that variant creates its gradient tensor; the count is independent of whether a particular dummy input gives a numerically zero gradient. The first six rows were recomputed with PyTorch using `scripts/count_ablation_parameters.py`; the canonical row comes from `conditioning_source_ablation/model.py` and its recorded run metadata. Thus equal total counts do not imply equal effective capacity. The 16 static-routing coordinates are inactive in the four dynamically routed rows, the direct-feature projection is inactive in routing-only, and the controller is inactive in static routing.

Table 8: Total/forward-active trainable parameters for the earlier six-variant ablation, followed by the clean canonical reference. IMDb counts exclude the shared $42{,}958 \times 128$ token embedding; including it adds $5{,}498{,}624$ active parameters.

Variant	pSMNIST	Noise-CIFAR	PTB	IMDb (no emb.)
Dynamic + direct feature	60,717/60,701	91,117/91,101	140,373/140,357	100,325/100,309
Time only	60,717/60,701	91,117/91,101	140,373/140,357	100,325/100,309
Current input only	60,717/60,701	91,117/91,101	140,373/140,357	100,325/100,309
Routing only (earlier cell)	60,717/59,613	91,117/90,013	140,373/138,181	100,325/99,221
Backbone only	60,717/60,717	91,117/91,117	140,373/140,373	100,325/100,325
Static routing	60,717/45,725	91,117/57,885	140,373/114,885	100,325/60,949
Canonical ICGR	59,098/59,098	89,498/89,498	137,666/137,666	98,706/98,706

The first six rows are counted from the earlier dual-path development implementation. Its routing-only row disables the direct controller-feature projection but still evaluates a 512-weight rank-two correction, its one scalar correction scale, and two Laplacian scales. These $512 + 1 + 2 = 515$ active parameters are absent from the clean canonical `GraphMix` implementation in `conditioning_source_ablation/`; the two cells are therefore different forward maps, and we make no checkpoint-conversion or numerical-equivalence claim. The canonical row is the source for the rounded ICGR counts in Tables 1–3 and for the source and operator controls.

A.4 Extended six-variant mechanism ablation

The earlier dual-path experiment instantiates six variants with the same parameter set, although the active parameters differ as reported above. Its routing-only variant uses the history controller only for routing. "Dynamic + direct feature" also projects it into the backbone; time-only and current-input replace its information source while retaining both paths; backbone-only makes routing static; and static routing removes both dynamic paths.

Table 9: Six-variant mechanism ablation under a common protocol (mean ± standard deviation, $n = 5$).

Variant	pSMNIST acc. (%)	Noise-CIFAR acc. (%)	PTB BPC ↓
Dynamic + direct feature	95.73 ± 0.29	60.34 ± 0.44	1.662 ± 0.002
Routing only (earlier cell)	$\mathbf{96.46 \pm 0.30}$	$\mathbf{60.40 \pm 0.25}$	$\mathbf{1.646 \pm 0.002}$
Time-only controller	93.91 ± 0.35	59.48 ± 0.24	1.811 ± 0.003
Current-input controller	94.03 ± 0.22	59.43 ± 0.37	1.692 ± 0.004
Backbone-only conditioning	93.56 ± 0.34	58.98 ± 0.28	1.678 ± 0.003
Static routing	93.91 ± 0.45	58.46 ± 0.34	1.731 ± 0.002

Adding the direct feature to dynamic routing is worse by 0.73 and 0.06 accuracy points on pSMNIST and noise-CIFAR and by 0.016 BPC on PTB; on IMDb the corresponding accuracies are $88.06 \pm 0.21\%$ and $87.64 \pm 0.24\%$. Within this development architecture, dynamic routing improves over backbone-only by 2.90 and 1.42 accuracy points and 0.032 BPC. Its routing-only variant improves over fully static routing by 2.55 and 1.94 points and 0.085 BPC, but that comparison changes both routing and the controller information path.

For the cleaner main-text source control, all three variants use the same routing-only canonical cell and instantiated parameter count, fixed graph, paired learned-parameter initialization, minibatch order, and optimization. Zero controller input in time-only and per-step state reset in current-input alter how some weights participate effectively; these are information-source controls, not a proof of equal effective capacity. The pSMNIST runs use 50 epochs, batch size 256, and warmup–cosine scheduling; PTB uses 60 epochs, batch size 128, and a factor-0.1 learning-rate drop at epoch 50. Checkpoints are selected on validation performance.

A.5 Complete routing-replay tables

Table 10: Complete fixed-length replay results, including agreement with the checkpoint's normal prediction.

Routing trajectory	Target acc. (%)	Donor acc. (%)	Agreement (%)
Normal	96.300 ± 0.111	—	100.000 ± 0.000
Self-replay	96.300 ± 0.111	—	100.000 ± 0.000
Same-class donor	96.319 ± 0.127	96.319 ± 0.127	93.011 ± 0.104
Different-class donor	0.757 ± 0.098	95.508 ± 0.286	1.316 ± 0.078

A.6 Class-template routing interventions

For each frozen checkpoint, class templates are estimated from the pSMNIST training split only. Because every routing coordinate is positive, we average in log-routing space:

$$\mu_y(t) = \mathbb{E}_{(x,y)\in\mathcal{D}_{\text{train}}} \left[\log(z_t(x) + \epsilon) \mid y\right], \qquad \epsilon = 10^{-6}. \tag{23}$$

The template for a designated class y_d is $z_t^{\text{template}} = \exp(\mu_{y_d}(t))$. The class-shift intervention applied to a target of class y_t is

$$z_t' = \exp\big(\log\big(z_t^{\text{self}} + \epsilon\big) - \mu_{y_t}(t) + \mu_{y_d}(t)\big)\,. \tag{24}$$

For wrong-class interventions, $y_d \neq y_t$ is supplied by the same ten deterministic different-class donor maps used for trajectory replay. Test labels are used only to construct these declared diagnostic interventions. We average maps within each checkpoint, then report the mean and sample standard deviation across the three independently trained checkpoints. Thus, the relatively large 4.302-point deviation for wrong-class-template designated accuracy is checkpoint-to-checkpoint variation, not variation over maps.

Table 11: Class-template interventions by independently trained checkpoint. The target-class template is an oracle diagnostic because the correct class selects the template. Other rows first average ten designated-class maps within each checkpoint.

Checkpoint	Intervention	Target acc. (%)	Designated-class acc. (%)
0	Target-class template (oracle)	100.00	—
1	Target-class template (oracle)	98.26	—
2	Target-class template (oracle)	99.06	—
0	Wrong-class template	0.00	100.00
1	Wrong-class template	1.12	91.48
2	Wrong-class template	0.35	96.81
0	Class-shift	11.11	57.79
1	Class-shift	12.17	58.87
2	Class-shift	12.00	57.98

The target-class template obtains $99.107 \pm 0.871\%$ target accuracy. This is not a classifier result: the true class chooses the template. Likewise, designated-class accuracy measures compliance with the class supplied by the intervention, not identification of an unseen donor. Wrong-class templates show that a training-derived class-shared routing pattern is sufficient to redirect predictions under the frozen model without preserving a donor instance's trajectory. These constructed input–routing pairs do not establish that every naturally evaluated sample uses a single shared template mechanism. Class-shift is also not a unique decomposition into class and non-class components: its residual may retain class information, and the transformation can change numerical scale and compatibility with the evolving main state.

Table 12: Complete different-class segmented replay results, including normal-prediction agreement.

Donor duration	Position	Target acc. (%)	Donor acc. (%)	Agreement (%)
25%	Prefix 1:196	71.51 ± 1.36	8.40 ± 0.44	72.45 ± 1.41
	Suffix 589:784	88.22 ± 1.19	3.90 ± 0.39	90.26 ± 1.26
50%	Prefix 1:392	29.81 ± 1.60	38.66 ± 2.21	30.23 ± 1.60
	Suffix 393:784	43.58 ± 3.74	23.32 ± 2.30	44.42 ± 3.78
75%	Prefix 1:588	4.69 ± 0.52	85.84 ± 1.08	5.21 ± 0.52
	Suffix 197:784	10.51 ± 0.85	64.46 ± 2.59	11.04 ± 0.86

A.7 Additional mechanism protocols

Frozen routing probes. For each of the three trained history checkpoints, we freeze every ICGR parameter and fit linear classifiers under the same data split and validation-selection protocol. The final-state probe receives $c_T \in \mathbb{R}^{64}$. The trajectory probe receives the complete flattened $z_{1:T}$, with $T = 784$ and $K = 16$, rather than a temporally pooled or low-frequency representation. Its $12{,}544$ standardized input coordinates give the ten-class linear head $125{,}450$ trainable parameters. This is a decodability diagnostic on representations produced by trained ICGR, not a capacity-matched independently trained baseline.

Features are standardized with training-split statistics. Each probe uses Adam with learning rate 10^{-2}, zero weight decay, batch size 1024, and at most 300 epochs; the epoch is selected by validation accuracy. Three probe initializations are averaged within checkpoint before aggregation; only the independently trained ICGR checkpoints are statistical replicates.

Table 13: Frozen linear probes and complete-model accuracy by training checkpoint. Probe initializations are averaged within checkpoint; aggregate deviations use checkpoints as replicates.

Checkpoint	Final c_T probe (%)	Full $z_{1:T}$ probe (%)	ICGR (%)
0	44.17	95.13	96.18
1	43.06	95.33	96.32
2	44.45	95.30	96.40
Mean $\pm$ std.	43.893 ± 0.735	95.253 ± 0.107	96.300 ± 0.111

The complete-trajectory probe trails ICGR by 1.047 ± 0.052 points. This shows that class labels are highly linearly decodable from the supplied full-trajectory representation. It does not show that all time positions are necessary: z_T alone is already a nonlinear transformation of c_T, and the full probe may rely on a subset or a time-aggregated statistic. Nor does decodability show that ICGR uses the same linear decoder or that routing improves long-term retention of the target history.

Fixed-prior controls. The graph-seed rerun resamples only A_0 while holding five training seeds and all other settings fixed; its graph-seed-0 row is a separate checkpoint set from the headline run. The three tested ER-like draws span 0.08 accuracy points. This checks sensitivity within a small set of draws, not robustness to other graph families.

Table 14: Fixed-prior controls on pSMNIST (mean $\pm$ standard deviation over five paired training seeds in a separate rerun).

Fixed-prior condition	Test acc. (%)
Graph seed 0	96.32 ± 0.29
Graph seed 1	96.24 ± 0.32
Graph seed 2	96.30 ± 0.27
$A_0 = 0$	87.61 ± 0.41

A.8 Permuted sequential MNIST

Table 15: Source-versus-rerun pSMNIST comparison for configurations with a corresponding published result. Published values are transcribed from the cited sources without assigning a mean, best-run, or single-run interpretation when the source does not specify one; reruns are mean $\pm$ standard deviation over five runs.

Model / source configuration	Published (%)	Our rerun (%)
Antisym. RNN, $H = 128$ (Chang et al., 2019)	95.8	94.63 ± 0.34
Lipschitz RNN, Euler, $H = 128$ (Erichson et al., 2021)	96.3	94.81 ± 0.29
coRNN, $H = 128$ (Rusch & Mishra, 2021a)	96.6	93.77 ± 0.36
UniCORNN, $L = 3$, $H = 128$ (Rusch & Mishra, 2021b)	97.8	95.32 ± 0.45

The final width, epoch budget, learning rate, and model-specific settings for our reruns are given in Table 16. The Antisym. RNN and Lipschitz RNN rows use the respective authors' implementations, adapted to our training framework; the same adapted implementations are used for noise-padded CIFAR-10. The reported Lipschitz rerun uses Euler integration and is therefore paired only with the published Euler configuration. The coRNN and UniCORNN rows use the respective official implementations, adapted to the common framework by changing the sequence-batch interface, device-aware state initialization, and shared training/validation checkpoint selection.

The reruns do not reach the corresponding published values. We do not attribute these gaps to a particular cause without a matched implementation and protocol audit; equal model labels and hidden widths do not establish identical training or evaluation conditions.

Single-seed width check. A separate seed-0 pSMNIST diagnostic changes only the ICGR backbone width from $D_{\mathrm{bb}} = 64$ to $D_{\mathrm{bb}} = 128$ and obtains 97.89% test accuracy. The resulting model has $116{,}826$ trainable parameters (116.8k). All other data, model, graph, and optimization settings match the main ICGR recipe: $N = 128$, $K = 16$, $D_{\mathrm{ctrl}} = 64$, the same permutation, split, and graph construction, Adam with learning rate 10^{-3} and weight decay 10^{-5}, warmup–cosine scheduling, batch size 256, gradient clipping at 1, and a 50-epoch budget. As in the main runs, the checkpoint is selected by highest validation accuracy before one test evaluation. This capacity-changing, single-seed result is not included in the five-seed main comparison or in Table 15, and is not evidence of matching a published baseline under a common protocol.

All pSMNIST runs use cross-entropy loss and gradient clipping at 1. Table 16 gives the complete final optimization and model-specific settings for the rows reported in Table 1.

Table 16: Final pSMNIST recipes. Warmup–cosine denotes five epochs of linear warmup followed by cosine decay.

Model	Selected recipe
Antisym. RNN	Adam, lr $3{\times}10^{-3}$, warmup–cosine, 50 epochs, batch 256; $H = 128$, $\epsilon = 0.1$, $\gamma = 0.01$, initialization std. 1.0.
Lipschitz RNN	Adam, lr 10^{-3}, warmup–cosine, 50 epochs, batch 256; $H = 128$, $\epsilon = 0.01$, $\alpha = 1.0$, $\beta = 0.75$, $\gamma = 10^{-3}$, initialization std. $6/H$.
coRNN	Adam, lr 3×10^{-3}, warmup–cosine, 100 epochs, batch 256; $H = 128$, $\Delta t = 0.076$, $\gamma = 0.4$, $\epsilon = 8.0$.
UniCORNN	Adam, lr 10^{-3}, warmup–cosine, 50 epochs, batch 256; three layers of width 128, $\Delta t = 0.076$, $\alpha = 0$.
LSTM	Adam, lr 10^{-3}, warmup–cosine, 50 epochs, batch 256; one layer of width 128.
S4D	Adam, lr 10^{-3}, warmup–cosine, 50 epochs, batch 256; complex S4D with model width 128.
HyperRNN	Adam, lr 10^{-3}, warmup–cosine, 50 epochs, batch 256; main width 128, hypernetwork width 64, embedding width 16, initial scale 1.0, controller input from the main state, embedding-scale std. 0.01.
QORNN	Adam, lr 10^{-3}, warmup–cosine, 50 epochs, batch 256; $H = 128$, 8-bit weights, 5 Björck iterations, 5 power iterations.
HadamRNN	Adam, lr 10^{-3}, warmup–cosine, 100 epochs, batch 256; $H = 512$, 4-bit input/output weights.
ICGR	Adam, lr 10^{-3}, weight decay 10^{-5}, warmup–cosine, 50 epochs, batch 256; $N = 128$, $K = 16$, $D_{\mathrm{ctrl}} = 64$, $D_{\mathrm{bb}} = 64$.

A.9 NOISE-PADDED CIFAR-10

All models receive the same length-1000 inputs described in Section 5.2, with 968 standard-Gaussian padding vectors appended after the image rows. All rows use batch size 256, Adam, and weight decay 10^{-5} unless explicitly marked otherwise; gradient clipping is also 1 unless marked "none". Table 17 records the final optimizer schedule; StepLR(s, g) denotes step size s and multiplicative factor g.

Table 17: Final noise-padded CIFAR-10 optimization recipes. Warmup–cosine uses five warmup epochs.

Model	Selected recipe
coRNN	lr 7.5×10^{-3}, StepLR$(100, 0.1)$, gradient clipping 1, 120 epochs.
UniCORNN	lr 3.14×10^{-3}, StepLR$(50, 0.1)$, gradient clipping 1, 60 epochs.
Lipschitz RNN	lr 3×10^{-3}, StepLR$(90, 0.1)$, no gradient clipping, 100 epochs.
Antisym. RNN ($H = 256$)	lr 10^{-3}, StepLR$(50, 0.1)$, gradient clipping 1, 60 epochs.
NM-RNN	lr 10^{-3}, warmup–cosine, gradient clipping 1, 50 epochs.
QORNN	lr 10^{-3}, StepLR$(50, 0.1)$, no gradient clipping, 60 epochs.
HadamRNN ($H = 1024$)	lr 10^{-3}, StepLR$(50, 0.1)$, gradient clipping 1, 60 epochs.
ICGR	lr 10^{-3}, weight decay 10^{-5}, warmup–cosine, gradient clipping 1, 50 epochs, batch 256.

Model-internal parameters use the following final defaults: coRNN uses $(\Delta t, \gamma, \epsilon) = (0.034, 1.3, 12.7)$; three-layer UniCORNN uses $\Delta t = 0.126$ and $\alpha = 13$; Lipschitz RNN uses $\epsilon = 0.01$, $\beta = 0.75$, and initialization scale $6/H$; Antisym. RNN uses $\epsilon = 0.1$ and $\gamma = 0.01$; and HadamRNN uses 4-bit input/output weights. QORNN and Lipschitz RNN use no weight decay in this table; all other rows use the common 10^{-5} value.

A.10 IMDB

We split the supplied $25{,}000$ training reviews $80/20$ with seed 42 and keep the separate $25{,}000$-review test set. Text is lowercased, HTML is removed, alphanumeric tokens are retained, and sequences are truncated to 500 tokens. A training-only vocabulary with minimum frequency 2 contains $42{,}958$ entries; its 128-dimensional embedding is randomly initialized and trainable, with dropout 0.1.

The reported IMDb reruns use Adam, learning rate 10^{-3}, batch size 256, weight decay 10^{-5}, gradient clipping at 1, and at most 20 epochs with early-stopping patience 10. Five linear warmup epochs precede cosine decay. Recurrent width is 128 for every row. The implementation-default rows are instantiated from the versioned files under `imdb/models/` and `imdb/dynamic_baselines.py`/`imdb/orthogonal_baselines.py`; the key non-default settings are coRNN $(\Delta t, \gamma, \epsilon) = (0.054, 4.9, 4.8)$, NM-RNN $(\mathrm{hidden_dim}, \mathrm{ctrl_dim}, \mathrm{rank}) = (128, 64, 16)$, HyperRNN $(\mathrm{hidden_dim}, \mathrm{hyper_dim}, \mathrm{embedding_dim}) = (128, 64, 16)$, QORNN

with 8-bit weights and 5 Björck and 5 power iterations, and HadamRNN with 4-bit weights. Checkpoints are selected by validation accuracy.

A.11 Character-level Penn Treebank

Each PTB line is an independent sequence; we append an end-of-sentence token, reset model state between lines, truncate or pad to 150 positions, and mask padded targets. With $\mathcal{M}$ the non-padding targets, we report

$$\mathrm{BPC} = \frac{\sum_{t \in \mathcal{M}} -\log p(x_{t+1} \mid x_{\le t})}{|\mathcal{M}| \log 2}. \tag{25}$$

The main comparison in Table 3(b) uses Adam for 60 epochs with StepLR step size 50, multiplicative factor 0.1, and batch size 256. ICGR additionally uses weight decay 10^{-5} and gradient clipping at 1. The separate conditioning-source controls use batch size 128. Table 18 records the remaining settings, and all checkpoints are selected by validation BPC.

Table 18: Final character-level PTB recipes.

Model	Selected recipe
LSTM	sequence length 150, lr 10^{-3}; widths as reported in Table 3.
GRU	sequence length 150, lr 10^{-3}; widths as reported in Table 3.
NM-RNN	sequence length 150, lr 3×10^{-3}; $H = 256$.
QORNN	sequence length 150, lr 10^{-3}; 8-bit weights; widths as reported in Table 3.
HadamRNN	sequence length 150, lr 10^{-3}; 8-bit input/output weights, $H = 1024$.
S4D	sequence length 150, lr 10^{-3}; widths as reported in Table 3.
Mamba	sequence length 150, lr 10^{-3}; widths as reported in Table 3.
ICGR	sequence length 150, lr 10^{-3}; $N = 128$, $D_{\mathrm{ctrl}} = 64$, $D_{\mathrm{bb}} = 128$.

B Complete Theory and Proofs

This appendix proves the results in Section 4 for the discrete update in Section 3. The current input, controller state, and modulation are held fixed whenever a partial state Jacobian is taken. The proofs do not establish stability of the controller GRU, the optimizer, or a task-specific variant with a different forward map.

For the local analysis, write

$$\eta_k(h) = 1 - \gamma_k(h), \quad D_k(h) = \mathrm{diag}(\eta_k(h)), \quad F_k(h) = h + D_k(h)\big(\mathcal{H}_k(h) - h\big), \tag{26}$$

and abbreviate $D_k = D_k(h_k)$. For any vector v, let $D_v = \mathrm{diag}(v)$.

B.1 Structured routed operators

Proof of Proposition 1. Let $p_r = U_{:,r}$. Entrywise,

$$\Big[A_0 \odot \Big(U \,\mathrm{diag}(z_k^{(\ell)}) U^\top\Big)\Big]_{ij} = (A_0)_{ij} \sum_{r=1}^{K} z_{k,r}^{(\ell)} (p_r)_i (p_r)_j \tag{27}$$

$$= \left[\sum_{r=1}^{K} z_{k,r}^{(\ell)} D_{p_r} A_0 D_{p_r}\right]_{ij}, \tag{28}$$

which proves Equation (12). Each diagonal rescaling $D_{p_r} A_0 D_{p_r}$ is skew-symmetric because $A_0^\top = -A_0$, and it cannot create a nonzero entry outside the support of A_0. Their weighted sum therefore has the same two properties. Every real skew-symmetric matrix S satisfies $h^\top S h = 0$, since this scalar equals its transpose $h^\top S^\top h = -h^\top S h$.

Finally, suppress the sample superscript and set $s = h^\top D_k C_k h$. Since $D_k^\top = D_k$ and $C_k^\top = -C_k$,

$$s = s^\top = h^\top C_k^\top D_k h = -h^\top C_k D_k h. \tag{29}$$

Averaging this expression with the original one gives

$$h^\top D_k C_k h = \frac{1}{2} h^\top (D_k C_k - C_k D_k) h, \qquad |h^\top D_k C_k h| \le \frac{1}{2} \|D_k C_k - C_k D_k\|_2 \|h\|_2^2. \tag{30}$$

The matrix $D_k C_k - C_k D_k$ is symmetric, so the inequality follows from its Rayleigh quotient. □

Remark 1. *Consider only the frozen, unsaturated, graph-only linear substep with a scalar gate. Freeze $C = C_k$, set $D_k = \eta I$ with $0 \le \eta \le 1$, and omit the input write, saturation, correction, and gate dependence on h_k. The transition is*

$$B_\eta h = \big[(1-\eta) I + \eta C\big] h, \qquad C^\top = -C. \tag{31}$$

If the eigenvalues of C are $i\omega_j$, then

$$|\lambda_j(B_\eta)| = \sqrt{(1-\eta)^2 + \eta^2 \omega_j^2}, \qquad \|B_\eta\|_2^2 = (1-\eta)^2 + \eta^2 \|C\|_2^2, \tag{32}$$

and consequently

$$\|B_\eta\|_2 \le 1 \quad \Longleftrightarrow \quad \eta \le \min\left\{1, \frac{2}{1 + \|C\|_2^2}\right\}. \tag{33}$$

The modulus is neither identically one nor necessarily below one. Unlike an exponential or Cayley construction that uses a skew matrix as a generator, ICGR uses the skew-routed current inside a target. Skew-symmetry therefore does not by itself make even this isolated substep orthogonal or contractive, and it gives no gradient-stability guarantee for the full cell. These formulas do not apply to the actual coordinate-wise graph term $(I - D_k) + D_k C_k$, much less to the saturated, input-driven, corrected update with state-dependent D_k.

Corollary B.1. *Freeze D_k and C_k at one sequence position and isolate the graph-only target $\mathcal{H}_k(h) = C_k h$. For*

$$\widehat{F}_k(h) = (I - D_k) h + D_k C_k h, \qquad B_k = (I - D_k) D_k, \tag{34}$$

the exact one-step identity is

$$\|\widehat{F}_k(h)\|_2^2 = \|(I - D_k) h\|_2^2 + \|D_k C_k h\|_2^2 + h^\top (B_k C_k - C_k B_k) h. \tag{35}$$

If $D_k = \eta I$ with $0 \le \eta \le 1$, then the commutator term vanishes and, because C_k is normal,

$$\|(1-\eta) I + \eta C_k\|_2^2 = (1-\eta)^2 + \eta^2 \|C_k\|_2^2. \tag{36}$$

Thus this isolated linear substep is nonexpansive if and only if

$$0 \le \eta \le \min\left\{1, \frac{2}{1 + \|C_k\|_2^2}\right\}. \tag{37}$$

Proof. Expand the squared norm of $(I - D_k) h + D_k C_k h$. The cross term is

$$2 h^\top (I - D_k) D_k C_k h = h^\top (B_k C_k - C_k B_k) h \tag{38}$$

by the same transpose argument used in Proposition 1. If $D_k = \eta I$, then B_k is a scalar multiple of the identity and commutes with C_k. A real skew-symmetric matrix is normal, with eigenvalues $i\omega_j$, so the singular values of $(1-\eta) I + \eta C_k$ are $\sqrt{(1-\eta)^2 + \eta^2 \omega_j^2}$. Maximizing over j proves Equation (36). Its value is at most one precisely when $\eta[(1 + \|C_k\|_2^2)\eta - 2] \le 0$, which gives Equation (37). This corollary concerns only a frozen C_k, a scalar gate, a graph-only target, and an unsaturated linear substep. Its spectrum is not the spectrum of the actual coordinate-wise map $(I - D_k) + D_k C_k$. Input writes, drive saturation, the bounded correction, and the state dependence of D_k remain part of the full cell analyzed locally in Lemma B.2; for a fixed input/controller trajectory, C_k is fixed with respect to the current main state. □

B.2 Routing-mediated and trajectory sensitivity

Proof of Proposition 2. From Equation (12), with $B_r = D_{p_r} A_0 D_{p_r}$,

$$C(z)h = \sum_{r=1}^{K} z_r B_r h = [B_1 h \quad \cdots \quad B_K h]\, z = V(h) z. \tag{39}$$

Holding h and x fixed therefore gives

$$\frac{\partial \widetilde{q}}{\partial z} = V(h), \qquad \frac{\partial q}{\partial \widetilde{q}} = \operatorname{diag}\big[\operatorname{sech}^2(\widetilde{q}/\kappa)\big] = S. \tag{40}$$

The implemented update can be written as

$$F(h, x, z) = h + D_{\eta(q)}\big(q + \psi(q) - h\big), \tag{41}$$

where the dependence of both ψ and η on h is held fixed for this partial derivative. Differentiating with respect to q gives

$$\left.\frac{\partial F}{\partial q}\right|_h = D_\eta + D_\eta J_{\psi,q} + D_{\mathcal{H}-h} J_{\eta,q} = L. \tag{42}$$

Combining Equations (40) and (42) proves Equation (16).

Because $B_r^\top = -B_r$, each column satisfies $h^\top B_r h = 0$. If $h \neq 0$, all columns of $V(h)$ therefore lie in the $(N-1)$-dimensional subspace $h^\perp$, so

$$\operatorname{rank}(LSV(h)) \leq \operatorname{rank} V(h) \leq \min\{K, N-1\}. \tag{43}$$

If $V(h)(z - z') = 0$, then the two modulations give the same routed current, hence the same $\widetilde{q}$, q, backbone variables, correction, retention rate, and next state. This equality is exact rather than a linear approximation. Finally, $V(0) = 0$, so at zero main state the current routing modulation cannot affect the update, although the encoded input can still write through $Ex + b_E$. □

Proof of Corollary 4.1. Linearizing one step along the fixed trajectory gives

$$\delta h_{k+1} = J_k \delta h_k + G_k \delta z_k. \tag{44}$$

Starting from $\delta h_1 = 0$ and recursively substituting this identity yields

$$\delta h_{T+1} = G_T \delta z_T + J_T G_{T-1} \delta z_{T-1} + \cdots + J_T J_{T-1} \cdots J_2 G_1 \delta z_1, \tag{45}$$

which is Equation (18). The expression is the differential of the unrolled recurrence at the reference trajectory; a finite routing replacement additionally contains higher-order terms. □

Proof of Proposition 3. Coordinate-wise saturation and the bounded correction give $|\mathcal{H}_{k,i}| \leq \kappa + 1$. Hence

$$|h_{k+1,i}| \leq (1 - \eta_{k,i})|h_{k,i}| + \eta_{k,i}|\mathcal{H}_{k,i}| \leq \max\{|h_{k,i}|, \kappa + 1\}. \tag{46}$$

Induction from the initial state h_1 proves Equation (21). □

B.3 Sensitivity supplied by the direct drive

Proposition 4. *With h_k, x_k, and z_k fixed, regard the post-saturation drive $q_k \in \mathbb{R}^N$ in Equation (6) as independent and let the shared backbone width be D. For the adaptive-gate cell,*

$$\frac{\partial h_{k+1}}{\partial q_k} = D_k + R_k, \qquad \operatorname{rank}(R_k) \leq D. \tag{47}$$

If the additive q_k term is removed from the target, the corresponding drive Jacobian has rank at most D. When every $\eta_{k,i} > 0$,

$$\operatorname{rank}\left(\frac{\partial h_{k+1}}{\partial q_k}\right) \geq N - D. \tag{48}$$

Conditional on a frozen retention gate,

$$\frac{\partial h_{k+1}}{\partial q_k} = D_k(I + J_{\psi,q}), \qquad \operatorname{rank}(J_{\psi,q}) \leq D. \tag{49}$$

The direct drive supplies a diagonal sensitivity path that is not forced through the compact backbone. This result differentiates with respect to the implemented post-saturation variable q_k, not the controller state, input history, or z_k, so it does not show that the model preserves more historical information. At $N = 128$ and $D = 64$, the lower bound with the direct path and the upper bound without it both equal 64. For the raw drive $\widetilde{q}_k$, the chain rule gives

$$\frac{\partial h_{k+1}}{\partial \widetilde{q}_k} = (D_k + R_k)\, S_k, \qquad S_k = \operatorname{diag}\big(\operatorname{sech}^2(\widetilde{q}_k/\kappa)\big)\,. \tag{50}$$

Thus saturation attenuates the direct path coordinate-wise.

Proof of Proposition 4. Hold h_k, x_k, and z_k fixed and expose the shared D-dimensional backbone state as $\xi_k = \xi_k(q_k)$, where q_k is the post-saturation drive. Both ψ_k and η_k depend on q_k only through ξ_k. Differentiating

$$h_{k+1} = h_k + D_{\eta_k(q_k)}\big(q_k + \psi_k(q_k) - h_k\big) \tag{51}$$

gives

$$\frac{\partial h_{k+1}}{\partial q_k} = D_k + D_k J_{\psi,\xi} J_{\xi,q} + D_{\mathcal{H}_k - h_k} J_{\eta,\xi} J_{\xi,q} \tag{52}$$

$$= D_k + \left[D_k J_{\psi,\xi} + D_{\mathcal{H}_k - h_k} J_{\eta,\xi}\right] J_{\xi,q}. \tag{53}$$

The second term factors through $\mathbb{R}^D$, so it is an admissible R_k with rank at most D.

If the additive q_k is removed from the target, the entire derivative is

$$\left[D_k J_{\psi,\xi} + D_{\psi_k - h_k} J_{\eta,\xi}\right] J_{\xi,q}, \tag{54}$$

which again has rank at most D. If every $\eta_{k,i} > 0$, then D_k is nonsingular and the rank inequality $\operatorname{rank}(A + B) \geq \operatorname{rank}(A) - \operatorname{rank}(B)$ gives

$$\operatorname{rank}(D_k + R_k) \geq N - D. \tag{55}$$

Finally, freezing the gate sets $J_{\eta,q} = 0$ and leaves

$$\frac{\partial h_{k+1}}{\partial q_k} = D_k + D_k J_{\psi,q} = D_k(I + J_{\psi,q}), \tag{56}$$

where $J_{\psi,q} = J_{\psi,\xi} J_{\xi,q}$ has rank at most D. Finally, differentiating Equation (6) gives

$$\frac{\partial q_k}{\partial \widetilde{q}_k} = \operatorname{diag}\big(\operatorname{sech}^2(\widetilde{q}_k/\kappa)\big) = S_k. \tag{57}$$

Right-multiplying the post-saturation Jacobian by S_k proves Equation (50). □

B.4 Conditional local state-Jacobian bound

Lemma B.2. *Along any differentiable trajectory, let $J_k = \partial F_k/\partial h_k$, $J_{\mathcal{H},k} = \partial \mathcal{H}_k/\partial h_k$, and $J_{\eta,k} = \partial \eta_k/\partial h_k$. Then*

$$J_k = I + D_k(J_{\mathcal{H},k} - I) + D_{\mathcal{H}_k - h_k} J_{\eta,k}. \tag{58}$$

Define

$$\delta_k = \|\eta_k\|_\infty (1 + \|J_{\mathcal{H},k}\|_2) + \|\mathcal{H}_k - h_k\|_\infty \|J_{\eta,k}\|_2. \tag{59}$$

Then $\|J_k - I\|_2 \leq \delta_k$. If $\delta_k < 1$, every perturbation v satisfies

$$(1 - \delta_k)\|v\|_2 \leq \|J_k v\|_2 \leq (1 + \delta_k)\|v\|_2. \tag{60}$$

Moreover, if $\delta_j < 1$ for every $j = t, \ldots, T-1$ and $\mathcal{J}_{t:T} = J_{T-1} \cdots J_t$, then

$$\prod_{j=t}^{T-1}(1 - \delta_j) \leq \sigma_{\min}(\mathcal{J}_{t:T}) \leq \sigma_{\max}(\mathcal{J}_{t:T}) \leq \prod_{j=t}^{T-1}(1 + \delta_j). \tag{61}$$

The exponential parameterization gives

$$J_{\eta,k} = \Delta t\, D_{\gamma_k} J_{r,k}, \qquad \|\eta_k\|_\infty \le \Delta t\, \|r_k\|_\infty, \qquad \|J_{\eta,k}\|_2 \le \Delta t\, \|J_{r,k}\|_2. \tag{62}$$

Hence $\|J_k - I\|_2 = O(\Delta t)$ if r_k, $\mathcal{H}_k - h_k$, $J_{\mathcal{H},k}$, and $J_{r,k}$ remain uniformly bounded independently of step size. This does not apply when Equation (22) is recalibrated to hold the discrete half-life fixed, because the nominal rate then scales as $1/\Delta t$. The envelope is a generic conditional diagnostic, not evidence that the condition holds along a complete trajectory or that a long Jacobian product is well conditioned.

Proof of Lemma B.2. Differentiate Equation (26) at h_k. The derivative of the explicit state term is I, the derivative through the target is $D_k(J_{\mathcal{H},k} - I)$, and differentiating the coordinate-wise gate contributes $D_{\mathcal{H}_k - h_k} J_{\eta,k}$. This proves Equation (58). Therefore

$$\|J_k - I\|_2 \le \|D_k\|_2 \|J_{\mathcal{H},k} - I\|_2 + \|D_{\mathcal{H}_k - h_k}\|_2 \|J_{\eta,k}\|_2 \tag{63}$$

$$\le \|\eta_k\|_\infty (1 + \|J_{\mathcal{H},k}\|_2) + \|\mathcal{H}_k - h_k\|_\infty \|J_{\eta,k}\|_2 = \delta_k. \tag{64}$$

For every v, the triangle and reverse-triangle inequalities give

$$\|J_k v\|_2 \le (1 + \delta_k)\|v\|_2, \qquad \|J_k v\|_2 \ge (1 - \delta_k)\|v\|_2. \tag{65}$$

When $\delta_k < 1$, these are respectively upper and positive lower singular-value bounds. If the same condition holds at every step $j = t, \dots, T-1$, repeated use of

$$\sigma_{\max}(AB) \le \sigma_{\max}(A)\sigma_{\max}(B), \qquad \sigma_{\min}(AB) \ge \sigma_{\min}(A)\sigma_{\min}(B) \tag{66}$$

proves Equation (61). □

To prove the step-size statements, note that $r_{k,i} \ge 0$, so

$$\eta_{k,i} = 1 - e^{-\Delta t r_{k,i}} \le \Delta t r_{k,i}. \tag{67}$$

Moreover,

$$\frac{\partial \eta_{k,i}}{\partial r_{k,i}} = \Delta t e^{-\Delta t r_{k,i}} = \Delta t \gamma_{k,i}, \tag{68}$$

so the chain rule gives the first identity in Equation (62). Because $0 < \gamma_{k,i} \le 1$, the two norm bounds in that equation follow. Substitution into Equation (59) yields the explicit estimate

$$\delta_k \le \Delta t \left[\|r_k\|_\infty (1 + \|J_{\mathcal{H},k}\|_2) + \|\mathcal{H}_k - h_k\|_\infty \|J_{r,k}\|_2\right], \tag{69}$$

which proves the stated small-step scaling under the uniform boundedness conditions in Section 4.

More generally, none of the results above establishes global contraction of the nonlinear cell, stability of the joint controller–main-state system, or good conditioning of a long Jacobian product when the per-step hypotheses fail.