HIDDEN STATE-SPACE DECOMPOSITION FOR LONG-TERM RETENTION IN LIQUID TIME-CONSTANT NETWORKS

Anonymous authors
Paper under double-blind review

ABSTRACT

Continuous-time recurrent models naturally handle irregularly sampled data by evolving hidden states as a function of elapsed time. Among these models, Liquid Time-Constant (LTC) networks adapt the rate of state evolution based on the current input, allowing their dynamics to change over time. This adaptability comes with an inherent decay of their hidden state, where information is progressively attenuated from earlier observations. Writing new information and preserving earlier information depend on the same state dynamics. Rapid writing requires the state to change quickly, while long-term retention requires the state to decay slowly. We introduce State-Space Decomposed Liquid Time-Constant Networks (SSD-LTC), where we decompose the LTC hidden state into an adaptive liquid subspace and a controlled persistent subspace. The adaptive liquid component follows the LTC dynamics, while the persistent component uses an input-dependent write rate to balance writing new information with preserving its previous state. We theoretically characterize state attenuation and the writing–retention trade-off in LTCs, and establish boundedness and stability properties of SSD-LTC. Across ten regularly sampled and eight irregularly sampled benchmarks, SSD-LTC improves classification accuracy by 2.2 to 8.0 percentage points and reduces regression and forecasting errors by up to 30% compared with the baseline LTC. We further show through controlled recall and component-exchange experiments that the learned persistent component acts as memory. These results demonstrate that separating adaptive state evolution from persistent information retention improves temporal modeling while retaining the continuous-time dynamics of LTCs.

1 INTRODUCTION

Temporal data arise from observations of real-world systems that continuously evolve over time. Such data are widely encountered across physiological monitoring, human activity recognition, environmental forecasting, industrial process monitoring, autonomous systems, and financial forecasting. These observations often arrive at nonuniform intervals, producing irregularly sampled data (Biloš et al., 2023). The time between measurements can therefore affect how much the system changes before the next observation (Park et al., 2022; Chen et al., 2023). Continuous-time models capture this temporal evolution by maintaining an internal representation in the form of a hidden state whose dynamics evolve continuously between observations.

Recurrent neural networks (RNNs) are widely used to model temporal data by processing observations sequentially and carrying information from earlier observations through a hidden state (Elman, 1990). In standard RNNs, this state changes through discrete updates as new observations arrive. To model how the underlying dynamics evolve between observations, continuous-time models describe the hidden-state dynamics using ordinary differential equations (ODEs). Neural ODEs provide a general formulation in which these dynamics are learned and numerically integrated over time (Chen et al., 2018). As the hidden state evolves continuously, it must balance information retained from the past with changes induced by new input. The rate of decay determines how quickly previous information is lost as the state evolves. Continuous-time RNNs (CT-RNNs) use a fixed time constant, causing the hidden state to evolve and decay at a fixed rate (ichi Funahashi & Nakamura, 1993). Liquid time-constant (LTC) networks make the time constant dynamic by allowing it to vary with

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.

the current input and hidden state (Hasani et al., 2021). This allows the rate of state evolution and decay to adjust dynamically as the input changes.

LTCs naturally accommodate irregularly sampled observations by evolving the hidden state over the actual elapsed time between measurements. A numerical solver advances the state across each observation interval, allowing longer and shorter intervals to produce different state evolution. During this, the adaptive time constant allows the state to change rapidly for some inputs and more slowly for others. LTCs therefore combine elapsed-time evolution with input-dependent dynamics and bounded states, making them well suited to irregular time-series prediction (Hasani et al., 2021). However, continuous state evolution also introduces an inherent decay of the previous state. As the state evolves, information carried from earlier observations is progressively attenuated, with the amount of attenuation accumulating over time. Consequently, information that must persist across long intervals can become increasingly weak. In LTCs, the adaptive dynamics can change the rate of this decay, but they do not explicitly separate information that should be preserved from information that can be allowed to fade. We characterize this state attenuation in more detail in Section 2.

Recent works have addressed this attenuation by augmenting LTCs with associative memory. One approach couples an LTC with a Modern Hopfield Network (MHN), allowing previously stored states to be retrieved and used for prediction (Swain et al., 2026). A more recent method feeds the retrieved patterns back into the LTC dynamics to guide subsequent state evolution (Swain et al., 2027). A more detailled discussion on related works are discussed in Appendix A. In both the above cases, information that may be attenuated in the evolving LTC state is recovered through an external memory mechanism. This raises a more fundamental question: *can the LTC hidden state itself preserve useful information while continuing to adapt to new observations?*

Biological neural activity offers a useful perspective on this question. Human cortex exhibits a range of intrinsic timescales (Raut et al., 2020). Studies of working memory also find stable representations alongside changing activity in human cortex and macaque prefrontal cortex (Li & Curtis, 2023; Murray et al., 2017). These findings motivate separating the hidden state into directions that support rapidly changing and persistent information. Persistence alone is not sufficient: information must also be written quickly when new content arrives. A direction that changes slowly at all times preserves information, but also takes longer to store a new value.

We introduce *State-Space Decomposed Liquid Time-Constant Networks* (SSD-LTC), which separate the recurrent hidden state into two components with distinct dynamics. SSD-LTC learns a controlled persistent subspace and an adaptive liquid subspace that forms its orthogonal complement. At each observation, an LTC cell first produces a provisional hidden state, from which the adaptive liquid component is obtained. The persistent component is updated separately through *adaptive retention*. We use a controller to determine a candidate value and a write rate from the current input and previous adaptive liquid component. The write rate controls how quickly the persistent component moves toward the candidate: a high rate allows new information to be written quickly, while a rate approaching zero preserves the previous value without an intrinsic decay term. The update also accounts for the elapsed time between observations. The two components are then combined to form the next recurrent state, allowing persistent information to influence subsequent LTC dynamics.

We evaluate SSD-LTC on ten regularly sampled and eight irregularly sampled benchmarks. SSD-LTC shows consistent improvements across both regular and irregular tasks. It improves classification accuracy by 2.2 to 8.0 percentage points and reduces regression and forecasting errors by up to 30% compared with the baseline LTC. We further use a controlled delayed-recall task, adapted from denoising and selective copying (Jing et al., 2019; Gu & Dao, 2024), to examine whether the learned persistent component acts as memory. SSD-LTC achieves 41.2% exact recall at an unseen 128-step delay, while exchanging the persistent component between sequences transfers the recalled content.

Our main contributions are:

- We characterize how the inherent decay in LTC dynamics attenuates earlier state information and couples the incorporation of new information with the preservation of previous information (Section 2).
- We introduce SSD-LTC, which decomposes the LTC hidden state into adaptive liquid and controlled persistent subspaces, enabling selective writing and retention within a single recurrent state (Section 3).

- We establish theoretical properties of SSD-LTC, including rapid writing with long-term retention, bounded persistent and complete hidden states, and non-expansive persistent dynamics (Section 3.4 and Appendices C and D).
- We evaluate SSD-LTC on eighteen regularly and irregularly sampled benchmarks and demonstrate through delayed recall and component-exchange experiments that the learned persistent subspace acts as memory over long delays (Section 4).

2 HIDDEN-STATE ATTENUATION IN LTCS

We first examine how information from the previous hidden state is carried through an LTC update. Let $\{\boldsymbol{x}_t\}_{t=1}^{T}$ be a sequence of observations, where t indexes the observation and $\Delta_t > 0$ is the elapsed time between observations $t-1$ and t. Let $\boldsymbol{h}_t \in \mathbb{R}^{d_h}$ denote the LTC hidden state after processing $\boldsymbol{x}_t$, and let $i \in \{1, \ldots, d_h\}$ index a hidden unit.

The continuous-time LTC dynamics are governed by an ordinary differential equation. In practice, the state is advanced between observations using a discretized update. For hidden unit i, we use the semi-implicit LTC update of Hasani et al. (2021):

$$h_{t,i} = \frac{h_{t-1,i} + \Delta_t f_{t,i} A_i}{1 + \Delta_t(\tau_i^{-1} + f_{t,i})}, \tag{1}$$

where $\tau_i > 0$ is the intrinsic time constant, A_i is a learned target value, and $f_{t,i} \geq 0$ is the conductance determined by the input and hidden state.

The contribution of the previous state $h_{t-1,i}$ to $h_{t,i}$ is scaled by

$$\alpha_{t,i} = \frac{1}{1 + \Delta_t(\tau_i^{-1} + f_{t,i})} < 1. \tag{2}$$

The direct contribution of the previous state is attenuated at each subsequent observation. This attenuation remains even when $f_{t,i} = 0$, since the intrinsic decay τ_i^{-1} keeps $\alpha_{t,i} < 1$.

The conductance $f_{t,i}$ also controls how strongly new information affects the state. Increasing $f_{t,i}$ strengthens the drive toward A_i while decreasing $\alpha_{t,i}$, which reduces the direct contribution of $h_{t-1,i}$. Thus, writing new information and retaining the previous state are coupled through the same LTC dynamics. Proposition 2 in Appendix B formalizes this trade-off. It shows that, along the direct LTC state path, the time for which a written value can be retained relative to the time required to write it is bounded, even when the conductance varies over time. Slowing the state dynamics therefore cannot provide arbitrarily long retention while preserving rapid writing.

Equation 2 describes how the previous state is attenuated through the LTC update. LTCs do not provide an explicit mechanism to protect information from this attenuation caused by its internal decay dynamics. In Appendix B, we derive this attenuation from the continuous-time LTC dynamics and formally analyze the trade-off between incorporating new information and preserving previous information. Our proposed method addresses this limitation by separating adaptive state evolution and controlled retention into different subspaces of the hidden state.

3 METHODS

State-Space Decomposed Liquid Time-Constant Networks (SSD-LTC) divide a single recurrent hidden state into two learned, complementary subspaces with different dynamics. The *adaptive liquid subspace* follows the LTC dynamics, while the *controlled persistent subspace* uses adaptive retention to preserve information over longer intervals. We use a controller to determine a candidate value and a write rate for the persistent component, allowing its state to change quickly when new information is written and slowly when information is retained. This separation allows the hidden state to remain responsive to new observations while protecting persistent information from the attenuation described in Section 2. Figure 1 illustrates one recurrent step of SSD-LTC.

Given the previous recurrent state $\boldsymbol{h}_{t-1}$, current observation $\boldsymbol{x}_t$, and elapsed time Δ_t, we first apply the LTC update described in Section 2 and denote its output by $\widetilde{\boldsymbol{h}}_t$. This intermediate state is then decomposed into the adaptive liquid and controlled persistent subspaces.

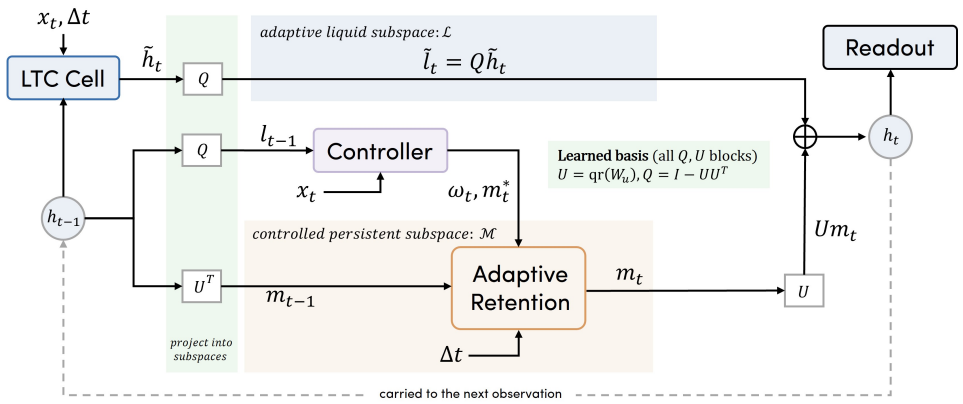


Figure 1: The proposed architecture for SSD-LTC. The LTC cell produces a provisional state $\widetilde{\boldsymbol{h}}_t$, which is projected onto the adaptive liquid subspace. The controller uses the previous liquid component and current input to determine a candidate value and write rate. Adaptive retention updates the persistent component over the elapsed time Δ_t. The adaptive liquid and controlled persistent components are then combined to form $\boldsymbol{h}_t$, which is passed to the readout and the next recurrent step. Here, $\boldsymbol{W}_u$ is the learned subspace parameter matrix, $\mathrm{qr}(\cdot)$ denotes the QR decomposition, $\boldsymbol{U}$ is the resulting orthonormal basis for the controlled persistent subspace.

3.1 Learned Complementary Subspaces

We decompose the hidden space into two orthogonal subspaces.

$$\mathbb{R}^{d_h} = \mathcal{L} \oplus \mathcal{M}, \qquad \dim(\mathcal{M}) = r, \qquad \dim(\mathcal{L}) = d_h - r, \tag{3}$$

where $0 \leq r < d_h$, d_h is the dimension of the hidden state, $\mathcal{M}$ is the controlled persistent subspace, and $\mathcal{L}$ is the adaptive liquid subspace.

The orientation of the persistent subspace is learned from a parameter matrix $\boldsymbol{W}_u \in \mathbb{R}^{d_h \times r}$. For $r > 0$, we apply a thin QR decomposition

$$(\boldsymbol{U}, \boldsymbol{R}_u) = \mathrm{qr}(\boldsymbol{W}_u), \qquad \boldsymbol{U}^\top \boldsymbol{U} = \boldsymbol{I}_r, \tag{4}$$

where the columns of $\boldsymbol{U} \in \mathbb{R}^{d_h \times r}$ form an orthonormal basis for $\mathcal{M}$. The persistent and liquid projection matrices are then

$$\boldsymbol{P} = \boldsymbol{U}\boldsymbol{U}^\top, \qquad \boldsymbol{Q} = \boldsymbol{I}_{d_h} - \boldsymbol{P}, \tag{5}$$

respectively. Since the two subspaces are orthogonal and complementary, any hidden state $\boldsymbol{h} \in \mathbb{R}^{d_h}$ can be written as

$$\boldsymbol{h} = \boldsymbol{Q}\boldsymbol{h} + \boldsymbol{U}\boldsymbol{m}, \qquad \boldsymbol{m} = \boldsymbol{U}^\top \boldsymbol{h} \in \mathbb{R}^r, \tag{6}$$

where $\boldsymbol{Q}\boldsymbol{h} \in \mathcal{L}$ is the adaptive liquid component, $\boldsymbol{U}\boldsymbol{m} \in \mathcal{M}$ is the controlled persistent component, and $\boldsymbol{m}$ contains its coordinates in the learned basis $\boldsymbol{U}$. At observation t, the previous adaptive liquid component is

$$\boldsymbol{\ell}_{t-1} = \boldsymbol{Q}\boldsymbol{h}_{t-1}, \tag{7}$$

which is provided to the controller together with the current input $\boldsymbol{x}_t$. The provisional LTC state $\widetilde{\boldsymbol{h}}_t$ is projected separately onto the adaptive liquid subspace:

$$\widetilde{\boldsymbol{\ell}}_t = \boldsymbol{Q}\widetilde{\boldsymbol{h}}_t = \widetilde{\boldsymbol{h}}_t - \boldsymbol{U}(\boldsymbol{U}^\top \widetilde{\boldsymbol{h}}_t). \tag{8}$$

This projected component retains the LTC dynamics in $\mathcal{L}$. The corresponding projection onto $\mathcal{M}$ is not used, since it would retain the decay dynamics described in Section 2. Instead, the persistent coordinates $\boldsymbol{m}_t$ are updated through adaptive retention, as described in the following sub-section.

3.2 Adaptive Retention Dynamics

We use a controller to determine how the persistent component is updated. The controller takes the previous adaptive liquid component $\boldsymbol{\ell}_{t-1}$ and the current observation $\boldsymbol{x}_t$ as input:

$$\boldsymbol{v}_t = [\boldsymbol{\ell}_{t-1}; \boldsymbol{x}_t] \in \mathbb{R}^{d_h+d_x}. \tag{9}$$

Two learned maps $g_\omega, g_c : \mathbb{R}^{d_h+d_x} \to \mathbb{R}^r$ produce a non-negative write rate $\boldsymbol{\omega}_t$ and a candidate value $\boldsymbol{m}_t^\star$ for each persistent coordinate:

$$\boldsymbol{\omega}_t = \mathrm{softplus}(g_\omega(\boldsymbol{v}_t)) \in \mathbb{R}^r_{\geq 0}, \tag{10}$$

$$\boldsymbol{m}_t^\star = \tanh(g_c(\boldsymbol{v}_t)) \in [-1, 1]^r. \tag{11}$$

Between two observations, the write rate and candidate value are held fixed over the interval $\xi \in [0, \Delta_t]$. The persistent coordinates evolve according to

$$\frac{d\boldsymbol{m}(\xi)}{d\xi} = \boldsymbol{\omega}_t \odot \big(\boldsymbol{m}_t^\star - \boldsymbol{m}(\xi)\big), \qquad \boldsymbol{m}(0) = \boldsymbol{m}_{t-1}, \qquad \boldsymbol{m}(\Delta_t) = \boldsymbol{m}_t. \tag{12}$$

Integrating equation 12 over the observation interval gives the retention factor

$$\boldsymbol{\rho}_t = \exp(-\boldsymbol{\omega}_t \Delta_t) \in (0, 1]^r, \tag{13}$$

which gives

$$\boldsymbol{m}_t = \boldsymbol{\rho}_t \odot \boldsymbol{m}_{t-1} + (\mathbf{1} - \boldsymbol{\rho}_t) \odot \boldsymbol{m}_t^\star. \tag{14}$$

The retention factor $\boldsymbol{\rho}_t$ determines how much of the previous persistent value is preserved. A small write rate keeps $\boldsymbol{\rho}_t$ close to one and preserves $\boldsymbol{m}_{t-1}$, whereas a larger write rate moves $\boldsymbol{m}_t$ more quickly toward the candidate $\boldsymbol{m}_t^\star$. Because $\boldsymbol{\rho}_t$ also depends on Δ_t, the update accounts directly for the elapsed time between observations.

3.3 State Reconstruction

The recurrent state is formed by combining the adaptive liquid component with the updated persistent component:

$$\boldsymbol{h}_t = \widetilde{\boldsymbol{\ell}}_t + \boldsymbol{U}\boldsymbol{m}_t. \tag{15}$$

Substituting equation 8 and equation 14 gives the SSD-LTC recurrence:

$$\boldsymbol{h}_t = \boldsymbol{Q}\widetilde{\boldsymbol{h}}_t + \boldsymbol{U}\left[\boldsymbol{\rho}_t \odot \boldsymbol{m}_{t-1} + (\mathbf{1} - \boldsymbol{\rho}_t) \odot \boldsymbol{m}_t^\star\right]. \tag{16}$$

We initialize the persistent coordinates as $\boldsymbol{m}_0 = \boldsymbol{U}^\top \boldsymbol{h}_0$, which gives $\boldsymbol{m}_0 = \mathbf{0}$ when the recurrent state is initialized to zero. The first term in equation 16 carries the adaptive LTC dynamics, while the second contains the persistent information updated through adaptive retention. SSD-LTC therefore learns both which directions of the hidden state form the controlled persistent subspace and how quickly information changes within this subspace.

3.4 Theoretical Properties of SSD-LTC

The SSD-LTC recurrence preserves the separation between the adaptive liquid and controlled persistent components:

$$\boldsymbol{U}^\top \boldsymbol{h}_t = \boldsymbol{m}_t, \qquad \boldsymbol{Q}\boldsymbol{h}_t = \boldsymbol{Q}\widetilde{\boldsymbol{h}}_t. \tag{17}$$

The adaptive liquid component follows the projected LTC dynamics, and the controlled persistent component follows the adaptive retention dynamics.

For persistent coordinate i, the fraction of the previous value retained over an observation interval is

$$\rho_{t,i} = \exp(-\omega_{t,i}\Delta_t). \tag{18}$$

A small write rate keeps $\rho_{t,i}$ close to one and preserves the previous value, while a larger write rate moves the coordinate toward its candidate. Since $\omega_{t,i} \geq 0$ has no positive lower bound, a persistent coordinate can change rapidly when new information is written and remain nearly unchanged when that information needs to persist.

The persistent component has no intrinsic decay, allowing its previous value to be preserved as the write rate approaches zero. The following proposition shows how SSD-LTC combines rapid writing with long-term persistence.

Table 1: Comparative results of various continuous-time models across regularly sampled benchmarks. The results are reported as mean and standard deviation over five seeds, with a 64-dimensional hidden state. ↑: higher is better; ↓: lower is better. The best result is in bold and the second best is underlined.

Dataset (metric)	LSTM	CT-RNN	Neural ODE	CT-GRU	LTC	LTC-MHN	AM-LTC	SSD-LTC
Gesture (Acc.↑)	63.21±0.61	57.81±1.31	45.99±2.53	66.86±1.62	69.45±1.14	71.23±1.10	<u>72.28</u>±1.10	**76.36**±1.53
Occupancy (Acc.↑)	91.17±1.89	92.57±0.52	88.29±1.22	89.59±1.54	93.66±0.19	<u>95.77</u>±0.22	95.25±0.22	**97.32**±0.20
Activity Recognition (Acc.↑)	93.75±0.22	93.60±0.44	94.19±0.25	94.15±0.26	94.51±0.61	<u>95.86</u>±0.44	95.14±0.35	**96.69**±0.35
Sequential MNIST (Acc.↑)	95.22±0.11	94.72±0.17	95.29±0.12	96.01±0.18	95.36±0.23	<u>97.30</u>±0.15	96.57±0.12	**98.19**±0.19
Ozone (Acc.↑)	93.42±0.27	84.87±0.13	91.23±0.04	95.37±0.28	95.61±0.12	<u>97.37</u>±0.18	96.91±0.14	**98.44**±0.23
UCI-HAR (Acc.↑)	79.75±0.10	80.13±0.07	82.90±0.16	83.27±0.12	84.22±0.05	86.63±0.04	<u>88.42</u>±0.04	**92.18**±0.03
Power (Sq. err.↓)	0.641±0.05	0.758±0.008	1.281±0.161	0.579±0.004	0.592±0.020	0.629±0.019	<u>0.518</u>±0.021	**0.412**±0.018
Traffic (MAE↓)	0.176±0.018	0.171±0.019	0.207±0.016	0.207±0.015	0.144±0.012	0.149±0.015	<u>0.141</u>±0.019	**0.115**±0.016
Electricity (MAE↓)	0.087±0.011	0.083±0.013	0.082±0.009	0.083±0.010	0.074±0.008	0.086±0.009	<u>0.068</u>±0.013	**0.059**±0.012
Weather (MAE↓)	0.344±0.031	0.337±0.028	0.359±0.033	0.333±0.026	<u>0.324</u>±0.022	0.339±0.028	0.334±0.022	**0.301**±0.025

Proposition 1 (Retention after rapid writing)**.** *Let I_w be an observation interval of length δ, followed by intervals I_h with total length T. For any $v \in (-1, 1)$ and $\varepsilon, \varepsilon' \in (0, 1)$, persistent coordinate i can write v during I_w and retain it across I_h by setting*

$$m^\star_{t,i} = v, \quad \omega_{t,i} \geq \frac{\log(1/\varepsilon)}{\delta} \ \text{ on } I_w, \qquad \sum_{k \in I_h} \omega_{k,i} \Delta_k \leq \log\left(\frac{1}{1-\varepsilon'}\right). \tag{19}$$

After I_w, the error relative to v is at most ε times its initial value. Across I_h, at least a fraction $1 - \varepsilon'$ of the written value is retained. These conditions can be satisfied for any $\delta, T > 0$, allowing the retention time to grow independently of the writing time.

Proposition 1 shows that the controlled persistent subspace can support both rapid writing and long-term retention. The candidate $m^\star_{t,i}$ determines what is written, while the write rate $\omega_{t,i}$ controls how quickly the persistent coordinate changes. Since the write rate can approach zero after new information is written, the same coordinate can remain close to its stored value for an arbitrarily long interval. This gives $\mathcal{M}$ the capacity to act as memory. We further examine whether the learned persistent subspace retains information over long delays in practice in Section 4.4.1.

We provide the complete theoretical analysis in Appendices C and D. The results show that the two subspaces retain their respective dynamics, the persistent coordinates remain bounded, and their dynamics are non-expansive. As the write rate approaches zero, the persistent component approaches an identity mapping, allowing information to remain unchanged across successive observations. We further show that the complete SSD-LTC hidden state remains bounded when the observation intervals have a positive lower bound.

4 Experiments and Results

We evaluate SSD-LTC on ten regularly sampled tasks and eight irregularly sampled forecasting datasets. We first compare its performance with LTC and other temporal models, and then examine whether the learned persistent subspace acts as memory through delayed-recall experiments and ablations. Details of the benchmarks, baselines, and training protocol are provided in Appendix E.1.

4.1 Evaluation on Regularly Sampled Data

Table 1 compares SSD-LTC with recurrent and continuous-time models on ten regularly sampled datasets. SSD-LTC achieves the best mean performance on all ten tasks. Compared with LTC, it improves classification accuracy by 1.83–7.96 percentage points, with the largest gains on UCI-HAR (+7.96) and Gesture (+6.91). On the regression and forecasting tasks, SSD-LTC reduces the error over LTC by 30.4% on Power, 20.1% on Traffic, 20.3% on Electricity, and 7.1% on Weather. These results show that separating adaptive and persistent dynamics within the recurrent state provides consistent gains across classification, regression, and forecasting tasks.

Table 2: Comparative results of various continuous-time models across irregularly sampled benchmarks. The results are reported as mean and standard deviation of MSE over five seeds. Recurrent baselines are parameter-matched to SSD-LTC; the DLinear and tPatchGNN forecasting models keep their original architectures. Seq. len gives the range of sequence lengths in each dataset. The best result is in bold and the second best is underlined.

Dataset	Seq. len	LSTM	CT-RNN	Neural ODE	CT-GRU	LTC	DLinear	tPatchGNN	SSD-LTC
CESNET	264–314	0.8183±0.0338	0.7141±0.0331	<u>0.7123</u>±0.0013	0.7501±0.0331	0.7819±0.0060	0.7211±0.0035	0.7286±0.0048	**0.6806**±0.0022
ClusterTrace	8–394	2.0880±0.0737	1.8348±0.0082	1.9148±0.0555	1.7973±0.1950	2.0005±0.1214	<u>1.6433</u>±0.0108	1.7498±0.3409	**1.4190**±0.0629
EPA-Air	18–336	0.5246±0.0485	0.3898±0.0086	0.4034±0.0097	0.5517±0.0395	<u>0.3810</u>±0.0282	0.4056±0.0076	0.4390±0.0456	**0.3521**±0.0112
FNSPID	40–45	0.0431±0.0012	0.0408±0.0015	0.0380±0.0027	0.0373±0.0028	0.0367±0.0010	0.1490±0.0020	<u>0.0357</u>±0.0046	**0.0331**±0.0017
GDELT	10–351	1.3327±0.0008	1.3312±0.0033	1.3310±0.0013	1.3281±0.0012	<u>1.3273</u>±0.0004	1.3433±0.0005	1.3286±0.0007	**1.3197**±0.0003
ILINet	8–9	5.8758±0.8552	5.6665±0.2552	<u>5.6360</u>±0.0490	6.2929±0.1966	6.5806±0.6117	10.5677±0.2402	11.9764±5.5033	**5.4221**±0.1794
RepoHealth	4–62	2.5695±0.2323	2.1489±0.3211	<u>1.5798</u>±0.0343	2.1186±0.2490	2.1067±0.1098	2.1103±0.2353	1.5915±0.3307	**1.4744**±0.1631
StudentLife	54–62	0.6300±0.0080	0.5817±0.0009	0.5893±0.0014	0.5824±0.0077	0.5744±0.0049	0.6406±0.0018	<u>0.5696</u>±0.0015	**0.5547**±0.0019

Table 3: Ablation study of the proposed SSD-LTC components. Results are reported as mean ± standard deviation over three seeds. Each variant removes or replaces one component of SSD-LTC. LTC results are repeated from Tables 1 and 2 for reference, while SSD-LTC and its variants are trained using the same seeds.

Variant	Gesture (Acc.↑)	UCI-HAR (Acc.↑)	CESNET (MSE↓)
SSD-LTC	$\mathbf{76.38 \pm 1.32}$	$\mathbf{92.17 \pm 0.03}$	$\mathbf{0.6809 \pm 0.0021}$
Fixed coordinate basis	73.06 ± 1.67	91.72 ± 0.04	0.7007 ± 0.0025
Constant write rate	72.85 ± 0.95	90.97 ± 0.04	0.7852 ± 0.0024
Euler retention step	71.91 ± 1.62	89.24 ± 0.06	0.7088 ± 0.0027
LTC	69.45 ± 1.14	84.22 ± 0.05	0.7819 ± 0.0060

4.2 Evaluation on Irregularly Sampled Data

Table 2 compares SSD-LTC with recurrent, continuous-time, and forecasting models on eight irregularly sampled datasets. SSD-LTC achieves the lowest mean MSE on all eight datasets. Compared with LTC, it reduces the error on every dataset, with particularly large reductions on RepoHealth (30.0%), ClusterTrace (29.1%), and ILINet (17.6%). Improvements are also observed on CESNET (13.0%), FNSPID (9.8%), EPA-Air (7.6%), StudentLife (3.4%), and GDELT (0.6%).

The datasets vary considerably in sequence length, from 4 observations in RepoHealth to 394 in ClusterTrace. The largest improvements over LTC are observed on RepoHealth (30.0%, 4–62 observations) and ClusterTrace (29.1%, 8–394 observations), despite their different sequence-length ranges. ILINet, with only 8–9 observations, also shows a 17.6% reduction. These results show that the improvements of SSD-LTC are observed across both short and long irregularly sampled sequences.

4.3 Ablation Study

Table 3 examines three components of SSD-LTC on Gesture ($T = 32$), UCI-HAR ($T = 128$), and CESNET, with $d_h = 64$ and $r = 16$. The *fixed coordinate basis* replaces the learned QR basis with $\boldsymbol{U} = [\boldsymbol{I}_r; \mathbf{0}]$, removing the learned orientation of the persistent subspace. The *constant write rate* replaces the input-dependent write rate with a learned fixed vector. The *Euler retention step* replaces the exact adaptive retention update with a forward Euler approximation, $\boldsymbol{\rho}_t = \mathbf{1} - \boldsymbol{\omega}_t \Delta_t$.

Performance decreases when any of the three components is removed. Fixing the persistent basis reduces accuracy by 3.32 percentage points on Gesture and 0.45 on UCI-HAR, and increases MSE by 2.9% on CESNET. Using a constant write rate reduces accuracy by 3.53 and 1.20 percentage points on Gesture and UCI-HAR, respectively, and increases MSE by 15.3% on CESNET. The larger degradation on CESNET highlights the importance of adapting the write rate when observation intervals vary. Despite these reductions, the fixed-basis variant remains better than LTC on all three tasks, showing that separating the persistent and adaptive liquid components is useful even without learning the orientation of the persistent subspace.

Table 4: Write and retention behavior on the delayed-recall task at the unseen delay $K = 128$. A cue is a marked input whose value must be stored and recalled after the delay, while distractors are the intervening inputs that should not overwrite the stored cue. The write fraction, $1 - \rho_{t,i}$, measures how much each persistent coordinate is updated at the cue or distractor and is averaged over persistent coordinates. Retained over delay is the product of $\rho_{t,i}$ across the delay and measures the fraction of the persistent state carried to recall. Results are averaged over three seeds.

	Write fraction (%)		Retained over	Exact
Write rate	at cue	at distractor	delay (%)	recall (%)
SSD-LTC (input-dependent)	39.9	0.16	81.5	41.2
Constant, learned	1.76	1.76	10.3	1.8
Constant, matched to distractor rate	0.16	0.16	81.5	0.1

Approximating adaptive retention with a forward Euler step reduces accuracy by 4.47 percentage points on Gesture and 2.93 on UCI-HAR, and increases MSE by 4.1% on CESNET. The Euler retention factor $1-\omega_{t,i}\Delta_t$ can become negative when $\omega_{t,i}\Delta_t > 1$, whereas the exact factor $\exp(-\omega_{t,i}\Delta_t)$ remains in $(0, 1]$. The exact update therefore preserves the bounded retention behavior of the persistent component while also accounting for the elapsed time without discretization error. The computational cost and parameter overhead of SSD-LTC are reported in Appendix E.2.

4.4 Observations and Discussion

4.4.1 Controlled persistent subspace as memory

We test whether the learned persistent subspace $\mathcal{M}$ acts as memory using a delayed-recall task adapted from denoising and selective copying (Jing et al., 2019; Gu & Dao, 2024). Each sequence contains four marked cues that must be recalled in order after K distractors. Models are trained with $K \in \{16, 32, 64\}$ and evaluated at the unseen delay $K = 128$, where exact recall has a chance level of 0.39%. We use $d_h = 64$ and $r = 16$ and report results over three seeds. Two fixed-rate variants are included to separate input-dependent writing from slow retention: one learns a constant write rate, while the other uses the rate applied by SSD-LTC during distractors.

Selective writing and retention. Table 4 shows that SSD-LTC increases the write fraction from 0.16% during distractors to 39.9% when a cue appears, while retaining 81.5% of the persistent state across the 128-step delay. This results in 41.2% exact recall. The fixed-rate variants cannot achieve both behaviors: the learned constant rate loses most of the stored state over the delay, while the distractor-matched rate preserves the state but cannot write the cues effectively.

Interchange intervention. We further test whether the recalled information is carried by $\mathcal{M}$ using interchange interventions (Geiger et al., 2021). For 512 pairs of sequences with different cue symbols, we exchange either the persistent or adaptive liquid component during the delay and continue the remaining sequence unchanged. Exchanging the persistent component transfers the donor's recalled sequence in 41.1–41.2% of pairs, matching the 41.2% obtained by exchanging the complete hidden state. Exchanging the adaptive liquid component gives 0.0% transfer.

Together, the delayed-recall and interchange experiments show that the learned persistent subspace acts as memory. SSD-LTC writes cue information into $\mathcal{M}$, preserves it while distractors are processed, and later uses the stored content for recall.

4.5 Learned Write-Rate Analysis

Figure 2 examines the write rates learned by SSD-LTC on CESNET, RepoHealth, and Gesture. The persistent coordinates learn different update timescales across the three datasets (panel a). These differences are reflected in the write fractions (panel b): CESNET is dominated by small updates, RepoHealth uses moderate updates more frequently, and Gesture combines frequent small updates with occasional larger changes. The learned write rates also determine how long previous candidate values contribute to the persistent state (panel c), producing different retention profiles across the

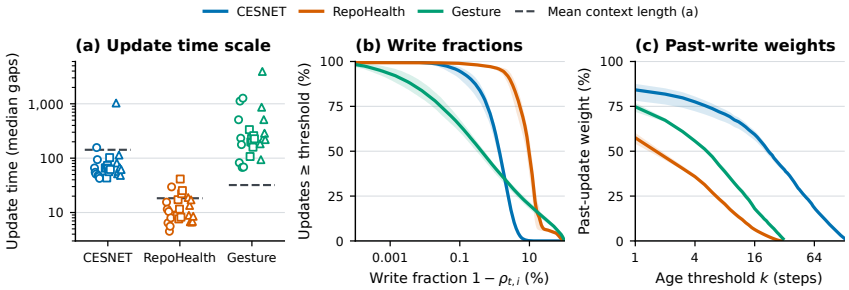


Figure 2: Learned adaptive retention on held-out CESNET, RepoHealth, and Gesture sequences. The models use $d_h = 64$ and $r = 8$, with three seeds per dataset. (a) Median persistent-coordinate update time $1/\omega_{t,i}$, measured in median observation gaps; dashed lines show mean context length. (b) Share of step–coordinate pairs above each write-fraction threshold $1 - \rho_{t,i}$. (c) Weight of candidate updates at least k steps old in the final persistent state. Panel (c) measures retention coefficients rather than the values stored. In panels (b) and (c), lines show seed means and shading spans the seeds.

three datasets. Together, these results show that SSD-LTC adapts both the magnitude and timescale of persistent-state updates to the temporal characteristics of each task.

4.6 Limitations

SSD-LTC has several limitations. First, it requires the dimension r of the persistent subspace to be specified in advance, and the appropriate balance between persistent and liquid dimensions may depend on the task. Second, the theoretical properties of the persistent dynamics do not imply that the learned subspace will always store task-relevant information; we establish this behavior empirically through controlled delayed-recall and interchange experiments. Third, our boundedness result for the complete hidden state also assumes that observation intervals are bounded away from zero. Finally, although SSD-LTC adds few parameters, learning and applying the subspace decomposition introduces additional computation compared with the underlying LTC.

5 Conclusion

In this work, we introduced State-Space Decomposed Liquid Time-Constant Networks (SSD-LTC) to address the attenuation of earlier information in LTC hidden states. Our analysis showed that LTC dynamics couple the incorporation of new information with the preservation of previous information through the same state evolution. SSD-LTC separates these roles within a single recurrent state by learning an adaptive liquid subspace that follows the LTC dynamics and a controlled persistent subspace with adaptive retention. This allows persistent coordinates to change rapidly when new information is written and remain nearly unchanged when information needs to be preserved. Across eighteen regularly and irregularly sampled benchmarks, SSD-LTC consistently improved over the baseline LTC and remained effective across different sequence lengths and sampling patterns. Delayed-recall experiments further showed that the persistent subspace selectively writes relevant information and retains it over unseen delays. Interchange interventions showed that transferring the persistent component also transfers the recalled content, providing evidence that the learned subspace acts as memory. Future work can extend state-space decomposition beyond LTCs to other continuous-time models and examine whether the same separation between adaptive and persistent dynamics remains effective. Broader evaluation across tasks, longer temporal horizons, and different forms of irregular sampling will also be important for understanding when persistent subspaces emerge and how reliably they function as memory.

AI USE STATEMENT

In this work, AI tools was used for editing the manuscript for grammar, improving readability and comprehension. We have not used any AI tools for model research, code, experiments or for interpretation of results. We have reviewed all AI-assisted work. We take responsibility for the final content of this work, including text, claims produced with the aid of generative AI.

REPRODUCIBILITY STATEMENT

We provide the model specification, theoretical derivations, and experimental details needed to reproduce this work. The complete SSD-LTC recurrence is described in Section 3, with assumptions and proofs for the theoretical results provided in Appendices C and D. Appendix E.1 details the benchmarks, baselines, training protocol, and evaluation settings, while Appendix E.2 reports implementation details and computational cost. Anonymized source code and instructions for reproducing the experiments will be included in the supplementary material.

REFERENCES

Y. Bengio, P. Simard, and P. Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166, 1994. doi: 10.1109/72.279181.

Marin Biloš, Kashif Rasul, Anderson Schneider, Yuriy Nevmyvaka, and Stephan Günnemann. Modeling temporal data as continuous functions with stochastic process diffusion. In *international conference on machine learning*, pp. 2452–2470. PMLR, 2023.

Ching Chang, Jeehyun Hwang, Yidan Shi, Haixin Wang, Wei Wang, Wen-Chih Peng, and Tien-Fu Chen. Time-IMM: A dataset and benchmark for irregular multimodal multivariate time series. In *Advances in Neural Information Processing Systems*, volume 38, 2025. doi: 10.52202/085713-1533.

Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. *Advances in neural information processing systems*, 31, 2018.

Yuqi Chen, Kan Ren, Yansen Wang, Yuchen Fang, Weiwei Sun, and Dongsheng Li. Contiformer: Continuous-time transformer for irregular time series modeling. *Advances in Neural Information Processing Systems*, 36:47143–47175, 2023.

Kyunghyun Cho, Bart Van Merriënboer, Çağlar Gulçehre, 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 (EMNLP)*, pp. 1724–1734, 2014.

Jeffrey L. Elman. Finding structure in time. *Cognitive Science*, 14(2):179–211, 1990. doi: 10.1207/S15516709COG1402_1.

Atticus Geiger, Hanson Lu, Thomas Icard, and Christopher Potts. Causal abstractions of neural networks. In *Advances in Neural Information Processing Systems*, volume 34, 2021.

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.

Ramin Hasani, Mathias Lechner, Alexander Amini, Daniela Rus, and Radu Grosu. Liquid time-constant networks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pp. 7657–7666, 2021.

Ramin Hasani, Mathias Lechner, Alexander Amini, Lucas Liebenwein, Aaron Ray, Max Tschaikowski, Gerald Teschl, and Daniela Rus. Closed-form continuous-time neural networks. *Nature Machine Intelligence*, 4(11):992–1003, 2022.

Ramin Hasani, Mathias Lechner, Tsun-Hsuan Wang, Makram Chahine, Alexander Amini, and Daniela Rus. Liquid structural state-space models. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=g4OTKRKfS7R.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8): 1735–1780, 1997.

Ken ichi Funahashi and Yuichi Nakamura. Approximation of dynamical systems by continuous time recurrent neural networks. *Neural Networks*, 6(6):801–806, 1993. ISSN 0893-6080. doi: https://doi.org/10.1016/S0893-6080(05)80125-X. URL https://www.sciencedirect.com/science/article/pii/S089360800580125X.

Li Jing, Caglar Gulcehre, John Peurifoy, Yichen Shen, Max Tegmark, Marin Soljačić, and Yoshua Bengio. Gated orthogonal recurrent units: On learning to forget. *Neural Computation*, 31(4): 765–783, 2019. doi: 10.1162/neco_a_01174.

Mathias Lechner and Ramin Hasani. Learning long-term dependencies in irregularly-sampled time series. *arXiv preprint arXiv:2006.04418*, 2020.

Hsin-Hung Li and Clayton E. Curtis. Neural population dynamics of human working memory. *Current Biology*, 33(17):3775–3784.e4, 2023. doi: 10.1016/j.cub.2023.07.067.

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

John D. Murray, Alberto Bernacchia, Nicholas A. Roy, Christos Constantinidis, Ranulfo Romo, and Xiao-Jing Wang. Stable population coding for working memory coexists with heterogeneous neural dynamics in prefrontal cortex. *Proceedings of the National Academy of Sciences*, 114(2): 394–399, 2017. doi: 10.1073/pnas.1619449114.

Daniel Neil, Michael Pfeiffer, and Shih-Chii Liu. Phased lstm: Accelerating recurrent network training for long or event-based sequences. *Advances in neural information processing systems*, 29, 2016.

Sung Woo Park, Kyungjae Lee, and Junseok Kwon. Neural markov controlled SDE: Stochastic optimization for continuous-time data. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=7DI6op61AY.

Ryan V. Raut, Abraham Z. Snyder, and Marcus E. Raichle. Hierarchical dynamics as a macroscopic organizing principle of the human brain. *Proceedings of the National Academy of Sciences*, 117 (34):20890–20897, 2020. doi: 10.1073/pnas.2003383117.

Yulia Rubanova, Ricky TQ Chen, and David K Duvenaud. Latent ordinary differential equations for irregularly-sampled time series. *Advances in neural information processing systems*, 32, 2019.

Bishal Ranjan Swain, Kyung Joo Cheoi, and Jaepil Ko. Coupling liquid time-constant encoders with modern hopfield memory. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 27409–27417, June 2026.

Bishal Ranjan Swain, Jaeyeon Hwang, and Jaepil Ko. Memory-induced attractor dynamics in liquid time-constant networks. In Luca Pasa, Alessandra Lintas, Igor V. Tetko, Alessio Micheli, Nicolò Navarin, Alessandro E.P. Villa, Domenico Tortorella, and Mirko Polato (eds.), *Artificial Neural Networks and Machine Learning – ICANN 2026*, pp. 167–178, Cham, 2027. Springer Nature Switzerland. ISBN 978-3-032-38401-0.

Ailing Zeng, Muxi Chen, Lei Zhang, and Qiang Xu. Are transformers effective for time series forecasting? In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pp. 11121–11128, 2023.

Weijia Zhang, Chenlong Yin, Hao Liu, Xiaofang Zhou, and Hui Xiong. Irregular multivariate time series forecasting: A transformable patching graph neural networks approach. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235, pp. 60179–60196, 2024.

A Related Work

A.1 From recurrent networks to gated memory

Elman networks introduced a recurrent hidden state that carries contextual information from one observation to the next (Elman, 1990). In its standard form, a recurrent neural network (RNN) updates $\boldsymbol{h}_t = \phi(\boldsymbol{W}_h \boldsymbol{h}_{t-1} + \boldsymbol{W}_x \boldsymbol{x}_t + \boldsymbol{b})$, where ϕ is an element-wise activation function, $\boldsymbol{W}_h$ and $\boldsymbol{W}_x$ are recurrent and input weight matrices, and $\boldsymbol{b}$ is a bias vector. This update compresses the complete input history into a single state. The same recurrent Jacobian is repeatedly composed through time, which makes long-range dependencies difficult to learn when its singular values cause gradients and state perturbations to vanish or grow (Bengio et al., 1994).

Long short-term memory (LSTM) introduced a separate cell state with multiplicative gates that regulate access to a nearly linear error path (Hochreiter & Schmidhuber, 1997). The gated recurrent unit (GRU) subsequently combined memory and exposed hidden state in a simpler update controlled by reset and update gates (Cho et al., 2014). These architectures learn when to retain or overwrite state and substantially improve optimization over long discrete sequences. Their recurrence is nevertheless indexed by observation order: elapsed physical time does not influence the state transition unless it is supplied through an additional mechanism.

A.2 Continuous-time recurrent models

Continuous-time RNNs (CT-RNNs) replace the discrete recurrence with an ODE, commonly written as

$$\boldsymbol{\tau} \odot \dot{\boldsymbol{h}}(t) = -\boldsymbol{h}(t) + \phi(\boldsymbol{W}_h \boldsymbol{h}(t) + \boldsymbol{W}_x \boldsymbol{x}(t) + \boldsymbol{b}), \tag{20}$$

where $\boldsymbol{\tau}$ fixes the intrinsic response rates of the hidden units (ichi Funahashi & Nakamura, 1993). Neural ODEs generalized this view by representing the full vector field with a neural network and evaluating the resulting flow with a numerical ODE solver (Chen et al., 2018). ODE-RNNs use such a flow between observations and a discrete recurrent update when an observation arrives, allowing the hidden state to evolve across arbitrary time gaps (Rubanova et al., 2019).

Other models incorporate time directly into gated recurrence. Phased LSTM adds an oscillatory time gate that opens the cell only during selected phases (Neil et al., 2016). CT-GRU interprets GRU gates as selecting among a bank of exponentially decaying traces with different time scales, with event timestamps determining their decay between observations (Mozer et al., 2017). These models improve the treatment of asynchronous events, but their temporal structure is supplied through fixed or separately parameterized clocks, decay traces, or ODE flows.

ODE-LSTM combines a continuous-time hidden-state flow with a separate LSTM-style memory path for irregularly sampled sequences (Lechner & Hasani, 2020). It shows a related separation of changing and persistent information. SSD-LTC studies that separation within a learned partition of one LTC hidden state.

A.3 Liquid time-constant networks

Liquid time-constant networks (LTCs) make the response rate itself dependent on the current input and hidden state (Hasani et al., 2021). Their dynamics are given in equation 21. The nonlinear conductance $\boldsymbol{f}$ therefore changes both the effective time constant and the input-dependent equilibrium. This gives LTCs adaptive continuous-time dynamics rather than the fixed decay of a CT-RNN. Closed-form continuous-time networks (CfCs) derive efficient approximate closed-form transitions from the LTC system (Hasani et al., 2022), while Liquid-S4 places input-dependent liquid transitions within a structured state-space model for long sequences (Hasani et al., 2023).

Selective state-space models offer another way to retain useful context. Mamba makes its state-space parameters depend on the input, allowing it to selectively propagate information along a sequence (Gu & Dao, 2024). SSD-LTC instead studies two update rules within one LTC state: projected adaptive liquid dynamics and controlled retention in persistent directions. The experiments compare the models listed in Appendix E.1.

B Analysis of State Attenuation in LTCs

Section 2 showed attenuation in the LTC state update. Here, we derive this behavior from the continuous-time LTC dynamics and formally analyze the relation between incorporating new information and preserving previous information.

The LTC hidden state evolves according to

$$\dot{\boldsymbol{h}}(t) = -\left[\boldsymbol{\tau}^{-1} + \boldsymbol{f}(\boldsymbol{h}(t), \boldsymbol{x}(t), t; \boldsymbol{\theta})\right] \odot \boldsymbol{h}(t) + \boldsymbol{f}(\boldsymbol{h}(t), \boldsymbol{x}(t), t; \boldsymbol{\theta}) \odot \boldsymbol{A}. \tag{21}$$

Here $\boldsymbol{h}(t) \in \mathbb{R}^{d_h}$ and $\boldsymbol{x}(t) \in \mathbb{R}^{d_x}$ denote the hidden state and input, $\boldsymbol{A} \in \mathbb{R}^{d_h}$ is a target vector, $\boldsymbol{\tau} \in \mathbb{R}^{d_h}_{>0}$ contains positive intrinsic time constants, and $\boldsymbol{\theta}$ denotes the learned parameters of the non-negative conductance map

$$\boldsymbol{f} : \mathbb{R}^{d_h} \times \mathbb{R}^{d_x} \times \mathbb{R} \longrightarrow \mathbb{R}^{d_h}_{\geq 0}. \tag{22}$$

The symbol $\odot$ denotes element-wise multiplication; vector reciprocals and vector exponentials are likewise evaluated element-wise throughout the paper. We next isolate the retention mechanism in equation 21. This analysis concerns the explicit state-carry path of an LTC. It does not assume that the complete nonlinear recurrent system is globally contractive.

B.1 Writing and Retention in LTCs

Consider one LTC coordinate with its conductance held fixed within each observation interval. The part of an earlier state carried directly from t_s to t_T is multiplied by

$$\mathcal{C}_i(T, s) = \prod_{k=s+1}^{T} e^{-(\tau_i^{-1} + f_{k,i})\Delta_k} = \exp\left[-\sum_{k=s+1}^{T} (\tau_i^{-1} + f_{k,i})\Delta_k\right]. \tag{23}$$

Since $f_{k,i} \geq 0$,

$$0 < \mathcal{C}_i(T, s) \leq \exp\left[-\frac{t_T - t_s}{\tau_i}\right]. \tag{24}$$

Thus the direct carry of an earlier state attenuates with elapsed time. When $f_{k,i} = 0$, the intrinsic decay rate τ_i^{-1} still attenuates that state. The scalar solution behind equation 23 is derived below.

The semi-implicit LTC update used in equation 1 has a similar form at each observation:

$$\widetilde{h}_{t,i} = \alpha_{t,i} h_{t-1,i} + \beta_{t,i} A_i, \tag{25}$$

where

$$\alpha_{t,i} = \frac{1}{1 + \Delta_t(\tau_i^{-1} + f_{t,i})}, \qquad \beta_{t,i} = \frac{\Delta_t f_{t,i}}{1 + \Delta_t(\tau_i^{-1} + f_{t,i})}. \tag{26}$$

Here $\alpha_{t,i}$ carries the previous value and $\beta_{t,i}$ controls the drive toward A_i. Increasing $f_{t,i}$ strengthens that drive while reducing direct retention. Decreasing it preserves more of the previous value while weakening the drive. The discrete carry coefficient and the derivatives of these terms are derived below. We now measure how this shared control affects writing and retention.

Both an LTC coordinate and the persistent coordinates introduced in Section 3 follow, along their explicit carry path, a relaxation

$$\dot{y}(\xi) = -\lambda(\xi)\big(y(\xi) - y^\star(\xi)\big), \qquad \lambda(\xi) \geq 0, \tag{27}$$

with rate $\lambda = \tau_i^{-1} + f_i$ and target $y^\star = f_i A_i / \lambda$ for LTC coordinate i. For an interval I, let $\mathcal{C}(I) = \exp[-\int_I \lambda(\xi)\, d\xi]$ denote its explicit carry coefficient, as in equation 23.

Definition 1 (Writing and retention)**.** Let $0 < \varepsilon, \varepsilon' < 1$. A coordinate following equation 27 *writes* a value v within time δ and then *retains* it for time T if there are consecutive intervals I_w and I_h of lengths δ and T such that (i) $y^\star \equiv v$ on I_w and $\mathcal{C}(I_w) \leq \varepsilon$, and (ii) $\mathcal{C}(I_h) \geq 1 - \varepsilon'$. The ratio T/δ is the *retention-to-write time ratio* for v.

Condition (i) overwrites the previous content: integrating equation 27 with a constant target gives $|y(\sup I_w) - v| = \mathcal{C}(I_w)\,|y(\inf I_w) - v| \leq \varepsilon\,|y(\inf I_w) - v|$. Condition (ii) keeps at least a fraction $1 - \varepsilon'$ of the written state on the explicit path, whatever targets are presented during the retention interval. A useful memory needs a short writing time and a long retention time, so it needs a large T/δ.

Proposition 2 (Limit on writing and retention in an LTC)**.**

(a) If $\lambda(\xi) \equiv \lambda$ is constant, then for every target schedule

$$\frac{T}{\delta} \leq \frac{\log\big(1/(1-\varepsilon')\big)}{\log(1/\varepsilon)}. \tag{28}$$

(b) Let LTC coordinate i, with $A_i \neq 0$, follow equation 21 along its explicit path for an arbitrary conductance schedule $f_i(\xi) \geq 0$. It can write v only if $0 \leq v/A_i < 1$, and whenever $1 - v/A_i \geq \kappa$ for some $\kappa \in (0, 1]$,

$$\frac{T}{\delta} \leq \frac{\log\big(1/(1-\varepsilon')\big)}{\kappa\, \log(1/\varepsilon)}, \tag{29}$$

independently of τ_i and of the conductance schedule.

For $\varepsilon = \varepsilon' = 0.1$, part (a) gives $T/\delta \leq 0.046$. Part (b) gives $T/\delta \leq 0.092$ when $\kappa = 1/2$. In both cases, the permitted retention time is less than a tenth of the writing time. With a fixed rate, slowing the update lengthens writing and retention by the same factor. In an LTC, the conductance sets both the value written and the rate of writing, while τ_i^{-1} continues to attenuate the previous value. The bound applies to graded values away from the saturation target A_i. Values near A_i can act as persistent flags. Appendix C provides the required proof.

Because $\boldsymbol{f}_t$ depends on $\boldsymbol{h}_{t-1}$, the full recurrent Jacobian also contains derivatives of $\boldsymbol{f}_t$ and cross-coordinate recurrent interactions. These terms can slow perturbations on a particular trajectory. The results show that the explicit state-carry term decays, and any longer-lived direction must emerge implicitly from the nonlinear dynamics. This observation motivates learned subspace construction in Section 3, where LTC dynamics are retained in one component and replaced by controlled, identity-approaching retention in another.

B.2 Supporting derivations

To derive equation 23, hold the conductance of coordinate i fixed at $f_i \geq 0$ over an interval of length Δ. Its dynamics then become

$$\dot{h}_i(t) = -\lambda_i h_i(t) + f_i A_i, \qquad \lambda_i = \tau_i^{-1} + f_i > 0. \tag{30}$$

The exact solution is

$$h_i(t+\Delta) = e^{-\lambda_i \Delta} h_i(t) + \big(1 - e^{-\lambda_i \Delta}\big)\, \frac{f_i A_i}{\lambda_i}. \tag{31}$$

Multiplying the direct carry factors across intervals gives equation 23 and equation 24.

For the semi-implicit LTC step in equation 25, hold the gates fixed when tracing an earlier state value. The direct carry coefficient is

$$\widetilde{\mathcal{C}}_i(T, s) = \prod_{k=s+1}^{T} \alpha_{k,i} = \exp\left[- \sum_{k=s+1}^{T} \log\big(1 + \Delta_k(\tau_i^{-1} + f_{k,i})\big) \right]. \tag{32}$$

With constant Δ and f_i, this becomes

$$\widetilde{\mathcal{C}}_i(T, s) = \big[1 + \Delta(\tau_i^{-1} + f_i)\big]^{-(T-s)}. \tag{33}$$

It approaches equation 23 as $\Delta \to 0$ at fixed elapsed time. The effective time constant is

$$\tau_{\mathrm{eff},t,i} = \frac{1}{\tau_i^{-1} + f_{t,i}}. \tag{34}$$

For fixed τ_i and Δ_t, differentiating the coefficients in equation 26 gives

$$\frac{\partial \alpha_{t,i}}{\partial f_{t,i}} = -\frac{\Delta_t}{\big[1 + \Delta_t(\tau_i^{-1} + f_{t,i})\big]^2} < 0, \qquad \frac{\partial \beta_{t,i}}{\partial f_{t,i}} = \frac{\Delta_t(1 + \Delta_t \tau_i^{-1})}{\big[1 + \Delta_t(\tau_i^{-1} + f_{t,i})\big]^2} > 0. \tag{35}$$

The conductance therefore controls both the drive toward A_i and the direct retention of the previous value.

C ADDITIONAL PROPERTIES AND PROOFS

This section proves Proposition 2, states and proves five additional structural properties, and proves Proposition 1. All derivatives hold the input $\boldsymbol{x}_t$, elapsed time Δ_t, basis $\boldsymbol{U}$, and learned parameters fixed.

Proof of Proposition 2. *(a)* With a constant rate, $\mathcal{C}(I_w) = e^{-\lambda\delta}$ and $\mathcal{C}(I_h) = e^{-\lambda T}$. Condition (i) of Definition 1 gives $\lambda\delta \geq \log(1/\varepsilon) > 0$, so $\lambda > 0$, and condition (ii) gives $\lambda T \leq \log(1/(1-\varepsilon'))$. Dividing the second inequality by the first proves equation 28.

(b) Along its explicit path, equation 30 is the relaxation equation 27 with $\lambda_i = \tau_i^{-1} + f_i$ and $y^\star = f_i A_i/(\tau_i^{-1} + f_i)$. The map $f \mapsto f/(\tau_i^{-1} + f)$ is strictly increasing on $[0, \infty)$, with derivative $\tau_i^{-1}/(\tau_i^{-1} + f)^2 > 0$, and has range $[0, 1)$. Hence $y^\star \equiv v$ on I_w requires $0 \leq v/A_i < 1$ and a constant conductance f_w on I_w satisfying $v/A_i = f_w/(\tau_i^{-1} + f_w)$. The corresponding write rate is

$$\lambda_w = \tau_i^{-1} + f_w = \frac{\tau_i^{-1}}{1 - v/A_i} \leq \frac{1}{\kappa\,\tau_i}. \tag{36}$$

Condition (i) gives $e^{-\lambda_w\delta} \leq \varepsilon$, so $\delta \geq \log(1/\varepsilon)/\lambda_w \geq \kappa\,\tau_i \log(1/\varepsilon)$. During the hold, $f_i \geq 0$ implies $\lambda_i \geq \tau_i^{-1}$, so $\mathcal{C}(I_h) \leq e^{-T/\tau_i}$ for every conductance schedule. Condition (ii) then gives $T \leq \tau_i \log(1/(1-\varepsilon'))$. Dividing the two bounds cancels τ_i and proves equation 29. □

Proposition 3 (Exact subspace replacement)**.** *For every step of equation 16,*

$$\boldsymbol{U}^\top \boldsymbol{h}_t = \boldsymbol{m}_t, \qquad \boldsymbol{Q}\boldsymbol{h}_t = \boldsymbol{Q}\widetilde{\boldsymbol{h}}_t. \tag{37}$$

Therefore the provisional LTC dynamics are preserved exactly in $\mathcal{L}$ and replaced exactly in $\mathcal{M}$, rather than being added to an auxiliary memory state. Indeed, left-multiplying equation 16 by $\boldsymbol{U}^\top$ and $\boldsymbol{Q}$, respectively, and using $\boldsymbol{U}^\top\boldsymbol{Q} = \boldsymbol{0}$, $\boldsymbol{U}^\top\boldsymbol{U} = \boldsymbol{I}_r$, $\boldsymbol{Q}^2 = \boldsymbol{Q}$, and $\boldsymbol{Q}\boldsymbol{U} = \boldsymbol{0}$ gives equation 37.

Proof of Proposition 3. Using $\boldsymbol{U}^\top\boldsymbol{Q} = \boldsymbol{0}$ and $\boldsymbol{U}^\top\boldsymbol{U} = \boldsymbol{I}_r$ in equation 15 gives

$$\boldsymbol{U}^\top\boldsymbol{h}_t = \boldsymbol{U}^\top\boldsymbol{Q}\widetilde{\boldsymbol{h}}_t + \boldsymbol{U}^\top\boldsymbol{U}\boldsymbol{m}_t = \boldsymbol{m}_t. \tag{38}$$

Similarly, $\boldsymbol{Q}^2 = \boldsymbol{Q}$ and $\boldsymbol{Q}\boldsymbol{U} = \boldsymbol{0}$ imply

$$\boldsymbol{Q}\boldsymbol{h}_t = \boldsymbol{Q}^2\widetilde{\boldsymbol{h}}_t + \boldsymbol{Q}\boldsymbol{U}\boldsymbol{m}_t = \boldsymbol{Q}\widetilde{\boldsymbol{h}}_t. \tag{39}$$

Thus the two components are replaced and retained exactly in their respective orthogonal subspaces. □

Proposition 4 (Exact-time and semigroup consistency)**.** *For fixed $\boldsymbol{\omega}$ and candidate $\boldsymbol{m}^\star$ over an interval, define*

$$\Phi_\Delta(\boldsymbol{m}) = \boldsymbol{m}^\star + \exp(-\boldsymbol{\omega}\Delta) \odot (\boldsymbol{m} - \boldsymbol{m}^\star). \tag{40}$$

Then equation 40 is the exact solution map of equation 12, and for any $\Delta_1, \Delta_2 \geq 0$,

$$\Phi_{\Delta_2}(\Phi_{\Delta_1}(\boldsymbol{m})) = \Phi_{\Delta_1+\Delta_2}(\boldsymbol{m}). \tag{41}$$

To see this, subtract $\boldsymbol{m}^\star$ in equation 12; each resulting coordinate satisfies a homogeneous first-order ODE. Composing two solution maps multiplies their retention factors, and $e^{-\boldsymbol{\omega}\Delta_2} \odot e^{-\boldsymbol{\omega}\Delta_1} = e^{-\boldsymbol{\omega}(\Delta_1+\Delta_2)}$. Thus the memory branch handles arbitrary elapsed times without numerical integration error; the liquid branch retains the discretization of the underlying LTC update in equation 1. The semigroup identity assumes that the controller outputs are held fixed across the subdivision; recomputing them at an artificial intermediate observation defines a different controlled flow.

Proof of Proposition 4. For fixed $\boldsymbol{\omega}$ and $\boldsymbol{m}^\star$, let $\boldsymbol{z}(\xi) = \boldsymbol{m}(\xi) - \boldsymbol{m}^\star$. Equation 12 becomes

$$\frac{d\boldsymbol{z}(\xi)}{d\xi} = -\boldsymbol{\omega} \odot \boldsymbol{z}(\xi), \tag{42}$$

whose coordinate-wise solution is $\boldsymbol{z}(\Delta) = \exp(-\boldsymbol{\omega}\Delta) \odot \boldsymbol{z}(0)$. Restoring $\boldsymbol{m}^\star$ yields equation 40. Applying this solution twice and using the element-wise exponential identity gives

$$\Phi_{\Delta_2}(\Phi_{\Delta_1}(\boldsymbol{m})) - \boldsymbol{m}^\star = \exp(-\boldsymbol{\omega}\Delta_2) \odot \exp(-\boldsymbol{\omega}\Delta_1) \odot (\boldsymbol{m} - \boldsymbol{m}^\star) \tag{43}$$

$$= \exp[-\boldsymbol{\omega}(\Delta_1 + \Delta_2)] \odot (\boldsymbol{m} - \boldsymbol{m}^\star), \tag{44}$$

which proves equation 41. The identity requires the same controller outputs on both subintervals. □

Proposition 5 (Bounded protected coordinates)**.** *If $\boldsymbol{m}_0 \in [-1, 1]^r$, then*

$$\boldsymbol{m}_t \in [-1, 1]^r \quad \textit{for every } t, \qquad \|\boldsymbol{m}_t\|_\infty \le 1. \tag{45}$$

More generally, $\|\boldsymbol{m}_t\|_\infty \le \max\{\|\boldsymbol{m}_0\|_\infty, 1\}$.

Because $0 < \rho_{t,i} \le 1$ and $m^\star_{t,i} \in [-1, 1]$, each coordinate in equation 14 is a convex combination of its previous value and a bounded candidate. The claim therefore follows by induction. This guarantee applies to the protected coordinates; it does not by itself bound the liquid component or prove length generalization.

Proof of Proposition 5. For every coordinate, $0 < \rho_{t,i} \le 1$ and $m^\star_{t,i} \in [-1, 1]$. Hence

$$m_{t,i} = \rho_{t,i} m_{t-1,i} + (1 - \rho_{t,i}) m^\star_{t,i} \tag{46}$$

is a convex combination of $m_{t-1,i}$ and $m^\star_{t,i}$. If $m_{t-1,i} \in [-1, 1]$, then $m_{t,i} \in [-1, 1]$; induction from $t = 0$ proves equation 45. For an arbitrary initialization, the same argument shows

$$|m_{t,i}| \le \max\{|m_{0,i}|, 1\}, \tag{47}$$

and taking the maximum over coordinates gives the stated general bound. □

Proposition 6 (Controlled protected Jacobian)**.** *Let*

$$\boldsymbol{J}_t = \frac{\partial \boldsymbol{h}_t}{\partial \boldsymbol{h}_{t-1}}, \tag{48}$$

holding $\boldsymbol{U}$, $\boldsymbol{x}_t$, Δ_t, and all learned parameters, including $\boldsymbol{\theta}$, fixed. Since the memory controllers receive $\boldsymbol{Q}\boldsymbol{h}_{t-1}$ and $\boldsymbol{x}_t$, but not $\boldsymbol{U}^\top \boldsymbol{h}_{t-1}$ directly, the protected-to-protected block is

$$\boldsymbol{U}^\top \boldsymbol{J}_t \boldsymbol{U} = \operatorname{diag}(\boldsymbol{\rho}_t). \tag{49}$$

Consequently, the explicit protected-coordinate carry from step s to step T is

$$\boldsymbol{C}^{\mathcal{M}}_{T:s} = \prod_{k=s+1}^{T} \operatorname{diag}(\boldsymbol{\rho}_k) = \operatorname{diag}\left(\exp\left[- \sum_{k=s+1}^{T} \boldsymbol{\omega}_k \Delta_k \right] \right). \tag{50}$$

For a perturbation $\boldsymbol{U}\boldsymbol{\delta}$, $\boldsymbol{Q}(\boldsymbol{h}_{t-1} + \boldsymbol{U}\boldsymbol{\delta}) = \boldsymbol{Q}\boldsymbol{h}_{t-1}$; hence the controller outputs remain fixed while $\boldsymbol{m}_{t-1}$ changes by $\boldsymbol{\delta}$. The memory update then changes $\boldsymbol{m}_t$ by $\boldsymbol{\rho}_t \odot \boldsymbol{\delta}$ to first order, and Proposition 3 gives equation 49. This carry is governed by the dedicated memory rate rather than the LTC carry coefficient in equation 26. Equation 50 isolates the direct protected path, not the complete multi-step full-state Jacobian: memory can affect future liquid dynamics through the recurrent LTC cell, and the liquid state can subsequently alter later controller outputs.

Proof of Proposition 6. Write the memory update as

$$\boldsymbol{m}_t = G_t(\boldsymbol{m}_{t-1}, \boldsymbol{Q}\boldsymbol{h}_{t-1}), \tag{51}$$

where dependence on the fixed $\boldsymbol{x}_t$ and Δ_t is suppressed. Because the controllers depend on $\boldsymbol{Q}\boldsymbol{h}_{t-1}$ but not directly on $\boldsymbol{m}_{t-1} = \boldsymbol{U}^\top \boldsymbol{h}_{t-1}$,

$$\frac{\partial G_t}{\partial \boldsymbol{m}_{t-1}} = \operatorname{diag}(\boldsymbol{\rho}_t). \tag{52}$$

Let $F_t(\boldsymbol{h}_{t-1}) = \widetilde{\boldsymbol{h}}_t$ denote the provisional LTC map. The chain rule applied to $\boldsymbol{h}_t = \boldsymbol{Q}F_t(\boldsymbol{h}_{t-1}) + \boldsymbol{U}G_t$ gives

$$\boldsymbol{J}_t = \boldsymbol{Q}\frac{\partial F_t}{\partial \boldsymbol{h}_{t-1}} + \boldsymbol{U}\left[\operatorname{diag}(\boldsymbol{\rho}_t)\boldsymbol{U}^\top + \frac{\partial G_t}{\partial(\boldsymbol{Q}\boldsymbol{h}_{t-1})}\boldsymbol{Q}\right]. \tag{53}$$

Left-multiplying by $\boldsymbol{U}^\top$ and right-multiplying by $\boldsymbol{U}$, then using $\boldsymbol{U}^\top\boldsymbol{Q} = \boldsymbol{0}$ and $\boldsymbol{Q}\boldsymbol{U} = \boldsymbol{0}$, proves equation 49. Since each factor in equation 50 is diagonal, their product is the diagonal matrix of products of retention factors, and $\prod_k \exp(-\boldsymbol{\omega}_k\Delta_k) = \exp(-\sum_k \boldsymbol{\omega}_k\Delta_k)$ element-wise.

For completeness, let $\boldsymbol{V} \in \mathbb{R}^{d_h\times(d_h-r)}$ be an orthonormal basis of $\mathcal{L}$, so that $\boldsymbol{V}\boldsymbol{V}^\top = \boldsymbol{Q}$ and $[\boldsymbol{V}\ \boldsymbol{U}]$ is orthogonal. Define the original LTC Jacobian and the liquid-to-memory controller Jacobian by

$$\boldsymbol{K}_t = \frac{\partial F_t}{\partial \boldsymbol{h}_{t-1}}, \qquad \boldsymbol{H}_t = \frac{\partial G_t}{\partial(\boldsymbol{Q}\boldsymbol{h}_{t-1})}. \tag{54}$$

In the coordinates $[\boldsymbol{V}^\top\boldsymbol{h}; \boldsymbol{U}^\top\boldsymbol{h}]$, the complete one-step state Jacobian is therefore

$$\overline{\boldsymbol{J}}_t = \begin{bmatrix} \boldsymbol{V}^\top\boldsymbol{K}_t\boldsymbol{V} & \boldsymbol{V}^\top\boldsymbol{K}_t\boldsymbol{U} \\ \boldsymbol{H}_t\boldsymbol{V} & \operatorname{diag}(\boldsymbol{\rho}_t) \end{bmatrix}. \tag{55}$$

This expression keeps $\boldsymbol{K}_t$ as the Jacobian of the underlying LTC cell; the SSD construction does not require the LTC derivatives to be rederived.

The new lower-left block can be written explicitly. With $\boldsymbol{D}_{\rho,t} = \partial\boldsymbol{\rho}_t/\partial\boldsymbol{\ell}_{t-1}$ and $\boldsymbol{D}_{c,t} = \partial\boldsymbol{m}_t^\star/\partial\boldsymbol{\ell}_{t-1}$,

$$\boldsymbol{H}_t = \operatorname{diag}(\boldsymbol{m}_{t-1} - \boldsymbol{m}_t^\star)\boldsymbol{D}_{\rho,t} + \operatorname{diag}(\boldsymbol{1} - \boldsymbol{\rho}_t)\boldsymbol{D}_{c,t}, \qquad \boldsymbol{D}_{\rho,t} = -\Delta_t \operatorname{diag}(\boldsymbol{\rho}_t)\frac{\partial\boldsymbol{\omega}_t}{\partial\boldsymbol{\ell}_{t-1}}. \tag{56}$$

Thus the memory-to-memory block is controlled exactly by $\boldsymbol{\rho}_t$, whereas liquid perturbations can affect memory through the controller.

The complete multi-step Jacobian is

$$\frac{\partial[\boldsymbol{V}^\top\boldsymbol{h}_T; \boldsymbol{U}^\top\boldsymbol{h}_T]}{\partial[\boldsymbol{V}^\top\boldsymbol{h}_s; \boldsymbol{U}^\top\boldsymbol{h}_s]} = \overline{\boldsymbol{J}}_T\overline{\boldsymbol{J}}_{T-1}\cdots\overline{\boldsymbol{J}}_{s+1}. \tag{57}$$

Its protected-to-protected block contains both the direct product in equation 50 and indirect paths that leave and later re-enter $\mathcal{M}$. For example, over two steps this block contains

$$\underbrace{\boldsymbol{H}_t\boldsymbol{V}\boldsymbol{V}^\top\boldsymbol{K}_{t-1}\boldsymbol{U}}_{\mathcal{M}\to\mathcal{L}\to\mathcal{M}} + \underbrace{\operatorname{diag}(\boldsymbol{\rho}_t)\operatorname{diag}(\boldsymbol{\rho}_{t-1})}_{\text{direct protected path}}. \tag{58}$$

Consequently, multiplying the diagonal blocks along the protected path yields equation 50, but that product should not be identified with the full multi-step derivative. □

Because the softplus rate in equation 10 is strictly positive, the model never closes a memory coordinate exactly. The next property shows that it approaches an identity channel at a rate set by the write rate itself. Denote the controller pre-activations by $\boldsymbol{a}_t = g_\omega(\boldsymbol{v}_t)$ and $\boldsymbol{c}_t = g_c(\boldsymbol{v}_t)$, so that $\boldsymbol{\omega}_t = \operatorname{softplus}(\boldsymbol{a}_t)$ and $\boldsymbol{m}_t^\star = \tanh(\boldsymbol{c}_t)$, and let $\nabla_{\boldsymbol{\ell}}$ denote the gradient with respect to $\boldsymbol{\ell}_{t-1}$.

Proposition 7 (Near-identity protected channel)**.** *Suppose $\boldsymbol{m}_{t-1} \in [-1,1]^r$, which Proposition 5 guarantees whenever $\boldsymbol{m}_0 \in [-1,1]^r$. For every memory coordinate i, with $\boldsymbol{e}_i$ the i-th standard basis vector of $\mathbb{R}^r$,*

$$0 \le 1 - \rho_{t,i} \le \omega_{t,i}\Delta_t, \qquad |m_{t,i} - m_{t-1,i}| \le 2\,\omega_{t,i}\Delta_t, \tag{59}$$

$$\left\|\boldsymbol{e}_i^\top\boldsymbol{U}^\top\boldsymbol{J}_t\boldsymbol{Q}\right\|_2 \le \omega_{t,i}\Delta_t\left(2\|\nabla_{\boldsymbol{\ell}}a_{t,i}\|_2 + \|\nabla_{\boldsymbol{\ell}}c_{t,i}\|_2\right). \tag{60}$$

If the write is closed, $\boldsymbol{\omega}_t = \boldsymbol{0}$ independently of $\boldsymbol{v}_t$, then

$$\boldsymbol{U}^\top\boldsymbol{J}_t\boldsymbol{U} = \boldsymbol{I}_r, \qquad \boldsymbol{U}^\top\boldsymbol{J}_t\boldsymbol{Q} = \boldsymbol{0}, \qquad \boldsymbol{U}^\top\boldsymbol{h}_t = \boldsymbol{U}^\top\boldsymbol{h}_{t-1}. \tag{61}$$

As $\omega_{t,i}\Delta_t \to 0$, coordinate i therefore carries its previous value, its protected Jacobian entry approaches one, and its sensitivity to the liquid state vanishes, all at the same linear rate. For affine controllers, the gradients in equation 60 are fixed rows of the controller weight matrices, so the leakage from the liquid state into a slowly written coordinate is bounded by its write rate times a parameter-dependent constant. The closed-write identities equation 61 are the exact limit, realized when the rate is clamped to zero.

Proof of Proposition 7. State bounds. Since $1 - e^{-x} \le x$ for $x \ge 0$, $0 \le 1 - \rho_{t,i} = 1 - e^{-\omega_{t,i}\Delta_t} \le \omega_{t,i}\Delta_t$. From equation 14, $m_{t,i} - m_{t-1,i} = (1 - \rho_{t,i})(m^\star_{t,i} - m_{t-1,i})$, and $|m^\star_{t,i} - m_{t-1,i}| \le 2$ because both values lie in $[-1, 1]$. This proves equation 59.

Leakage bound. Right-multiplying equation 53 by $\boldsymbol{Q}$ and left-multiplying by $\boldsymbol{U}^\top$ gives $\boldsymbol{U}^\top \boldsymbol{J}_t \boldsymbol{Q} = \boldsymbol{H}_t \boldsymbol{Q}$. Since $\|\boldsymbol{Q}\|_2 \le 1$, $\|\boldsymbol{e}_i^\top \boldsymbol{U}^\top \boldsymbol{J}_t \boldsymbol{Q}\|_2 \le \|\boldsymbol{e}_i^\top \boldsymbol{H}_t\|_2$. The derivative of softplus is the logistic function σ, so $\partial \omega_{t,i}/\partial \boldsymbol{\ell}_{t-1} = \sigma(a_{t,i}) \nabla_{\boldsymbol{\ell}} a_{t,i}^\top$; similarly, $\partial m^\star_{t,i}/\partial \boldsymbol{\ell}_{t-1} = \big(1 - \tanh^2(c_{t,i})\big) \nabla_{\boldsymbol{\ell}} c_{t,i}^\top$. Row i of equation 56 is therefore

$$\boldsymbol{e}_i^\top \boldsymbol{H}_t = -\Delta_t \rho_{t,i}\, \sigma(a_{t,i})\, (m_{t-1,i} - m^\star_{t,i})\, \nabla_{\boldsymbol{\ell}} a_{t,i}^\top + (1 - \rho_{t,i}) \big(1 - \tanh^2(c_{t,i})\big) \nabla_{\boldsymbol{\ell}} c_{t,i}^\top. \tag{62}$$

The logistic function is bounded by softplus: with $y = e^a > 0$, $\sigma(a) = y/(1 + y) \le \log(1 + y) = \mathrm{softplus}(a)$, by the elementary inequality $\log(1+y) \ge y/(1+y)$. Hence $\sigma(a_{t,i}) \le \omega_{t,i}$. Combining this with $\rho_{t,i} \le 1$, $|m_{t-1,i} - m^\star_{t,i}| \le 2$, $1 - \tanh^2(c_{t,i}) \le 1$, and $1 - \rho_{t,i} \le \omega_{t,i}\Delta_t$ in equation 62 gives equation 60.

Closed write. If $\boldsymbol{\omega}_t = \boldsymbol{0}$ independently of $\boldsymbol{v}_t$, then $\boldsymbol{\rho}_t = \boldsymbol{1}$ and $G_t(\boldsymbol{m}_{t-1}, \boldsymbol{Q}\boldsymbol{h}_{t-1}) = \boldsymbol{m}_{t-1}$. Therefore $\partial G_t/\partial \boldsymbol{m}_{t-1} = \boldsymbol{I}_r$ and $\boldsymbol{H}_t = \boldsymbol{0}$, and equation 53 gives the first two identities in equation 61. Proposition 3 gives $\boldsymbol{U}^\top \boldsymbol{h}_t = \boldsymbol{m}_t = \boldsymbol{m}_{t-1} = \boldsymbol{U}^\top \boldsymbol{h}_{t-1}$. □

Proof of Proposition 1. On I_w the rate and target are constant, so $\mathcal{C}(I_w) = e^{-\omega_{t,i}\delta} \le \varepsilon$ by equation 19, and the target equals v; this is condition (i). Over I_h the carry coefficient is the product of the per-interval retention factors, $\mathcal{C}(I_h) = \exp(-\sum_{k \in I_h} \omega_{k,i}\Delta_k) \ge 1 - \varepsilon'$; this is condition (ii). Since $\tanh$ maps onto $(-1, 1)$ and softplus maps onto $(0, \infty)$, the choices $m^\star_{t,i} = v$, $\omega_{t,i} = \log(1/\varepsilon)/\delta$, and $\omega_{k,i} = \log(1/(1 - \varepsilon'))/T$ on I_h lie in the range of the controllers for every $\delta, T > 0$, so no bound on T/δ holds. For the drift, iterating equation 14 over I_h writes the final value as $\mathcal{C}(I_h)$ times the written value plus $1 - \mathcal{C}(I_h)$ times a convex combination of candidates in $[-1, 1]$. Both values lie in $[-1, 1]$, so their difference is at most 2, and the drift is at most $2\big(1 - \mathcal{C}(I_h)\big) \le 2\varepsilon'$. □

D Boundedness and stability of the complete state

The original LTC analysis bounds the effective time constants and neuronal states of an LTC trajectory under positivity and bounded-conductance conditions (Hasani et al., 2021). That result does not transfer to SSD-LTC directly: the LTC update equation 1 is applied to the reconstructed state $\boldsymbol{h}_{t-1} = \boldsymbol{Q}\widetilde{\boldsymbol{h}}_{t-1} + \boldsymbol{U}\boldsymbol{m}_{t-1}$, which is not an LTC trajectory, and an arbitrary learned basis $\boldsymbol{U}$ need not preserve the coordinate-wise intervals of the original theorem. We therefore bound the complete state directly from the one-step structure of the LTC update.

Lemma 1 (One-step LTC bound)**.** *Let $\tau_{\max} = \max_i \tau_i$ and suppose $\Delta_t \ge \Delta_{\min} > 0$ for every t. For any previous state $\boldsymbol{h}_{t-1}$, input $\boldsymbol{x}_t$, and non-negative conductance $\boldsymbol{f}_t$, the provisional state of equation 1 satisfies*

$$\|\widetilde{\boldsymbol{h}}_t\|_2 \le \bar{\alpha}\, \|\boldsymbol{h}_{t-1}\|_2 + \|\boldsymbol{A}\|_2, \qquad \bar{\alpha} = \frac{1}{1 + \Delta_{\min}/\tau_{\max}} < 1. \tag{63}$$

Proof. By equation 25, $\widetilde{h}_{t,i} = \alpha_{t,i} h_{t-1,i} + \beta_{t,i} A_i$. Since $f_{t,i} \ge 0$, $\alpha_{t,i} \le 1/(1 + \Delta_t \tau_i^{-1}) \le \bar{\alpha}$, and $0 \le \beta_{t,i} < 1$ by equation 26. Hence $|\widetilde{h}_{t,i}| \le \bar{\alpha}|h_{t-1,i}| + |A_i|$ for every coordinate. Because the Euclidean norm is monotone on non-negative vectors, the triangle inequality gives equation 63. □

The bound does not depend on the conductance network or its inputs: the conductance enters only through $\alpha_{t,i}$ and $\beta_{t,i}$, and non-negativity alone controls both.

Proposition 8 (Bounded complete state)**.** *Under the conditions of Lemma 1, let $M = \max\{\|\boldsymbol{m}_0\|_\infty, 1\}$ and $B = (\|\boldsymbol{A}\|_2 + \sqrt{r}\, M)/(1 - \bar{\alpha})$. Then, for every t,*

$$\|\boldsymbol{h}_t\|_2 \le \bar{\alpha}^{\,t} \|\boldsymbol{h}_0\|_2 + \big(1 - \bar{\alpha}^{\,t}\big)\, B \le \max\{\|\boldsymbol{h}_0\|_2, B\}. \tag{64}$$

Proof. Proposition 5 gives $\|\boldsymbol{m}_t\|_2 \le \sqrt{r}\, \|\boldsymbol{m}_t\|_\infty \le \sqrt{r}\, M$. Since $\|\boldsymbol{Q}\|_2 \le 1$ and $\|\boldsymbol{U}\boldsymbol{m}_t\|_2 = \|\boldsymbol{m}_t\|_2$, the reconstruction equation 15 and Lemma 1 give

$$\|\boldsymbol{h}_t\|_2 \le \|\boldsymbol{Q}\widetilde{\boldsymbol{h}}_t\|_2 + \|\boldsymbol{U}\boldsymbol{m}_t\|_2 \le \bar{\alpha}\, \|\boldsymbol{h}_{t-1}\|_2 + \|\boldsymbol{A}\|_2 + \sqrt{r}\, M. \tag{65}$$

Unrolling this recursion from $t = 0$ and summing the geometric series gives the first inequality in equation 64. The right-hand side is a convex combination of $\|\boldsymbol{h}_0\|_2$ and B, which gives the second. □

Remark. The bound scales as $\tau_{\max}/\Delta_{\min}$ and can therefore be loose for finely sampled sequences, but it is uniform in sequence length and in the trained conductance network. It uses the orthogonal decomposition only through $\|\boldsymbol{Q}\|_2 \leq 1$ and $\|\boldsymbol{U}\boldsymbol{m}_t\|_2 = \|\boldsymbol{m}_t\|_2$. The same argument applies to any LTC solver whose step writes each coordinate as $\alpha h_{t-1,i}$ plus a bounded drive with $\alpha \leq \bar{\alpha} < 1$, including solvers that unfold several such semi-implicit substeps per observation. It does not recover the coordinate-wise interval bounds of the original LTC theorem, which a learned rotation need not preserve.

Stability of the added memory dynamics. For two protected states driven by the same controller outputs, their difference $\boldsymbol{\delta}(\xi)$ obeys $d\boldsymbol{\delta}/d\xi = -\boldsymbol{\omega} \odot \boldsymbol{\delta}$. With $V = \frac{1}{2}\|\boldsymbol{\delta}\|_2^2$,

$$\frac{dV}{d\xi} = -\sum_{i=1}^{r} \omega_i \delta_i^2 \leq 0. \tag{66}$$

Equivalently, over one observation interval the exact update equation 14 gives $\boldsymbol{\delta}_t = \boldsymbol{\rho}_t \odot \boldsymbol{\delta}_{t-1}$, so $\|\boldsymbol{\delta}_t\|_2 \leq \|\boldsymbol{\delta}_{t-1}\|_2$. Hence the added subsystem introduces no positive-growth mode: it is non-expansive for $\boldsymbol{\omega} \geq \mathbf{0}$, exponentially contractive wherever the rates are bounded away from zero, and neutrally stable when a write is closed. The last case is precisely the identity-like regime used for long-term retention. This conditional statement does not assert global contraction of the complete coupled model, because different liquid trajectories can produce different controller outputs.

Together, these results show that SSD-LTC keeps the complete state bounded under the positivity conditions of the underlying LTC cell and a positive minimum observation gap, and that the added memory branch is bounded and non-expansive.

E Experimental details

E.1 Experimental setup

Benchmarks. We evaluate SSD-LTC on two groups of benchmarks. The first contains ten regularly sampled sequence tasks: six classification tasks (Gesture, Occupancy, Activity Recognition, Sequential MNIST, Ozone, and UCI-HAR), evaluated by accuracy, and four regression tasks (Power, Traffic, Electricity, and Weather), evaluated by squared error or mean absolute error. Gesture, Occupancy, Activity Recognition, Sequential MNIST, Ozone, Traffic, and Power follow the evaluation suite of Hasani et al. (2021). The second group contains eight irregularly sampled forecasting datasets (CESNET, ClusterTrace, EPA-Air, FNSPID, GDELT, ILINet, RepoHealth, and StudentLife), whose sequence lengths vary within each dataset (Table 2). These datasets come from the Time-IMM benchmark (Chang et al., 2025). We use their numerical modality and follow the dataset construction, splits, and forecasting protocol of that benchmark.

Baselines. We compare against a discrete gated recurrent network, LSTM (Hochreiter & Schmidhuber, 1997); continuous-time recurrent models, CT-RNN (ichi Funahashi & Nakamura, 1993), Neural ODE (Chen et al., 2018), CT-GRU (Mozer et al., 2017), and LTC (Hasani et al., 2021); and two LTCs augmented with external associative memories, LTC-MHN (Swain et al., 2026) and AM-LTC (Swain et al., 2027). On the irregular benchmarks we further include two forecasting models, DLinear (Zeng et al., 2023) and tPatchGNN (Zhang et al., 2024), which we retrain using the numerical-only implementations from Time-IMM (Chang et al., 2025).

Protocol. SSD-LTC uses a 64-dimensional hidden state throughout. The main benchmark, ablation, and delayed-recall results use $r = 16$. The sensitivity analysis in Appendix E.3 varies the rank. On the regularly sampled benchmarks, all recurrent baselines use a 64-dimensional hidden state. On the irregularly sampled benchmarks, we train every model for a fixed 200 epochs. We report test results for the checkpoint with the best validation performance. For the irregular benchmarks, LSTM, CT-RNN, Neural ODE, CT-GRU, and LTC are sized to approximately match the parameter count of SSD-LTC, about 26k parameters on CESNET. DLinear and tPatchGNN keep their original

architectures with adapters for irregular inputs. We report the mean and standard deviation over five independent training runs with different random seeds. Further information on model sizes and computational cost is reported in Section E.2.

E.2 Implementation and computational cost

The implementation never forms the dense projectors $\boldsymbol{P}$ or $\boldsymbol{Q}$. Equation 8 evaluates the projection using two matrix–vector products, with $O(d_h r)$ cost per time step. A thin QR factorization costs $O(d_h r^2)$ once per model forward pass, rather than once per sequence element. For affine g_ω and g_c, the memory controller adds $O(r(d_h + d_x))$ computation per time step and stores r recurrent memory coordinates. The rank r therefore controls both the capacity of the persistent subspace and the additional computation. At $r = 0$, the basis and memory controllers are omitted, $\boldsymbol{Q} = \boldsymbol{I}_{d_h}$, and equation 16 reduces exactly to the original LTC transition.

Table 5 reports model size and computational cost on the CESNET forecasting task, whose history and forecast spans are each seven days. Training cost uses the first training batch (32 windows, 316 padded positions, 9,250 observed events), and test cost uses the complete test split (22 windows, 314 padded positions, 6,273 observed events). All measurements use one NVIDIA RTX A6000.

Table 5: Model size and computational cost on CESNET. Epoch time covers training and validation. GFLOPs are estimated algorithmic operation counts for one training step, including the backward pass and optimizer update, and for one inference pass over the test split. Memory is the peak allocated GPU memory in MiB. Starred models use a 64-dimensional hidden state.

		Epoch	GFLOPs		Peak memory (MiB)	
Model	Parameters	time (s)	train batch	test split	train	test
LSTM	26,676	0.58	1.473	0.368	59.8	18.3
CT-RNN	26,506	2.70	8.051	1.845	155.3	18.3
Neural ODE	26,365	6.71	21.963	5.007	222.1	18.3
CT-GRU	25,984	1.37	1.683	0.391	77.5	18.4
LTC	26,397	4.56	6.246	1.423	3,406.8	19.3
DLinear	23,496	0.17	0.033	0.011	21.2	18.9
tPatchGNN	163,546	1.30	6.856	1.599	250.7	30.7
LTC-MHN*	157,506	4.11	53.510	12.201	4,489.6	744.4
AM-LTC*	47,938	7.04	7.010	1.615	3,215.1	27.2
SSD-LTC*	26,450	6.43	5.985	1.364	3,128.4	21.0

SSD-LTC and LTC have nearly identical parameter counts and estimated operation counts, and SSD-LTC uses slightly less peak training memory. Its epoch time is 1.4 times that of LTC. LSTM, CT-RNN, Neural ODE, CT-GRU, and LTC lie within 2% of the parameter count of SSD-LTC. LTC-MHN and AM-LTC use about 6.0 and 1.8 times as many parameters, respectively; tPatchGNN uses about 6.2 times as many.

Measurement details. Epoch time is the median over training runs of the per-run median epoch time in 200-epoch training logs. The server was shared, so timings are indicative rather than isolated benchmarks. Operation counts combine PyTorch's `FlopCounterMode`, which counts matrix, convolution, and attention operations, with a dispatch-level counter for element-wise operations and reductions, including those in the unrolled LTC and ODE solvers, in both the forward and backward passes. Each scalar addition, multiplication, division, or transcendental evaluation counts as one operation, and fused kernels (softmax, layer normalization, LSTM cells, QR, and attention) use explicit formulas. These are algorithmic estimates rather than measured hardware instructions. Peak memory is `torch.cuda.max_memory_allocated` during one optimizer step or one inference pass; it includes the model, inputs, intermediate tensors, gradients, and optimizer state, and excludes cached memory and the CUDA context.

E.3 SENSITIVITY TO PERSISTENT SUBSPACE RANK

We vary the persistent subspace rank while keeping the hidden-state dimension at $d_h = 64$. Table 6 reports results on Gesture, CESNET, and RepoHealth. The $r = 16$ row corresponds to the main results reported for these three datasets. It gives the highest Gesture accuracy and the lowest forecasting error on both CESNET and RepoHealth among the four ranks tested. Performance declines at $r = 32$ on all three datasets. Thus, increasing the persistent subspace rank does not consistently improve performance.

Table 6: Effect of persistent subspace rank with a 64-dimensional hidden state. Values are mean $\pm$ standard deviation. Gesture is measured by classification accuracy (%); CESNET and RepoHealth are measured by MSE. $\uparrow$: higher is better; $\downarrow$: lower is better.

Rank r	Gesture (Acc.$\uparrow$)	CESNET (MSE$\downarrow$)	RepoHealth (MSE$\downarrow$)
4	73.12 ± 1.62	0.7184 ± 0.0027	1.5949 ± 0.1728
8	75.12 ± 1.50	0.7245 ± 0.0048	1.5215 ± 0.1547
16	$\mathbf{76.36 \pm 1.53}$	$\mathbf{0.6806 \pm 0.0022}$	$\mathbf{1.4744 \pm 0.1631}$
32	71.90 ± 1.74	0.7702 ± 0.0080	1.7203 ± 0.1609