Continuity Laws for Sequential Models

Annan Yu*
Center for Applied Mathematics
Cornell University
Ithaca, NY 14853
ay262@cornell.edu

Dongwei Lyu
Department of EECS
University of California, Berkeley
Berkeley, CA 94720
dw_lyu@berkeley.edu

N. Benjamin Erichson
Lawrence Berkeley National Laboratory
Berkeley, CA 94720
erichson@lbl.gov

Abstract

Inductive biases influence the behavior and performance of sequential models. In this work, we study an underexplored inductive bias in sequential modeling: continuity in time. We ask a simple question: do models motivated by continuous-time formulations, such as state-space models, actually behave continuously in time, and does this translate into better performance on tasks with continuous temporal structure? To answer this, we formalize model continuity as convergence under temporal refinement, where a model is continuous if its predictions approach an underlying continuous trajectory as the temporal discretization is refined. We show that S4 exhibits stable continuous behavior, whereas S6 (the core of Mamba) can be more sensitive to input amplitude and selective dynamics, despite being derived from a continuous dynamical system. To study whether this distinction matters for learning, we also need a corresponding notion of task continuity. We therefore introduce a metric to quantify the continuity of datasets directly from their temporal structure. Across benchmarks, we find a clear empirical alignment between task continuity, model continuity, and model performance. Beyond an inductive bias, continuity also has practical consequences: we show that it enables a simple temporal subsampling strategy that improves both efficiency and performance.

1 Introduction

The recent surge of interest in sequential modeling, spanning language [53, 13, 6], time series [3, 11], and scientific data [33, 36, 34], has produced many new architectures that achieve strong performance on specific benchmarks, often at specific scales and under carefully chosen training recipes. However, benchmark performance alone gives only a partial view of model utility. It tells us what worked in one regime, but not why it worked, when it should continue to work, or when it is likely to fail. This is not a criticism of benchmarking; benchmarks are essential, but they are most useful when paired with principles that explain what is being measured. In more classical areas such as numerical analysis, algorithms are often judged not only by their performance on test problems, but also by the properties they preserve: stability, convergence, and sensitivity to discretization [32, 52]. For sequential models, we often lack analogous principles, making it hard to know whether a model has captured coherent structure in the data, or whether it is narrowly tuned to a particular temporal regime.

*Corresponding author.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

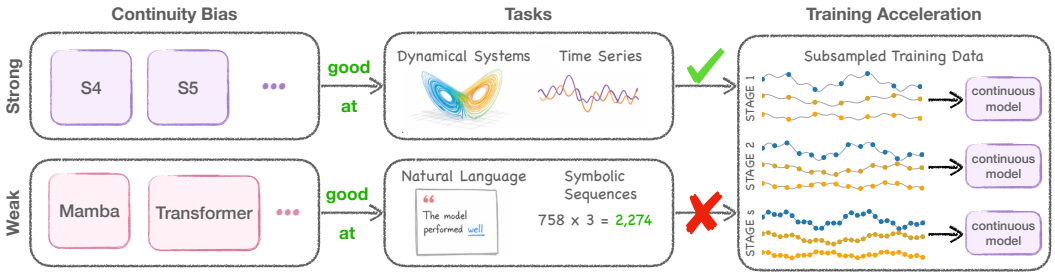


Figure 1: Continuity as a principle for model–task fit. Sequential models differ in how continuously they represent time, and tasks differ in how much continuous temporal structure they contain. Models with stronger temporal continuity are better matched to continuous problems such as dynamical systems and time series, while models with weaker continuity may be better matched to more discrete problems such as text or symbolic operations. When a model and task are both continuous, this structure can also be exploited algorithmically. An example is stage-wise training, where training begins on temporally subsampled data and progressively refines the resolution.

Neural scaling laws [28] provide one example of such a principle: they describe how performance changes as model and data size vary. While these laws are insightful, they answer only part of the question. Scaling laws tell us what may happen as a model becomes larger or sees more data; they say less about whether the model has the right structure for the problem in the first place. Scaling a mismatched architecture may improve performance, but it does not by itself make the architecture well-matched to the structure of the task. This matters especially in scientific and engineering applications, where data and compute are often limited, and where the structure of the problem is not arbitrary. In such settings, the question is not only "what can a sufficiently large model learn?" but "what does this model tend to learn?" This brings inductive biases to the forefront. For sequential models, several such biases are already familiar [60, 14], including frequency [59, 44], recency [27], locality [5], and multi-scale structure [10]. They help explain how a model processes time, memory, and variation across scales, but they do not exhaust the ways in which a model can be matched, or mismatched, to the temporal structure of a task. In this paper, we take a new perspective by asking:

Do models motivated by continuous-time formulations actually behave continuously in time, and does this help on tasks with continuous temporal structure?

On the model side, architectures differ in how they represent time. Transformers [53] operate through discrete token-to-token interactions. State-space models (SSMs) such as S4 [20], S5 [50], and S6 [18] take a different route: they are derived from continuous dynamical systems and are often described as continuous-time models. On the task side, it is intuitive that, for example, language is discrete and time series are continuous. By a *continuity law*, we mean a principle that characterizes the continuity of the model, the continuity of the task, and consequences when the two are aligned or misaligned. These ideas are made precise via three questions (see Figure 1):

- **What is a continuous model?** For models with an underlying continuous-time formulation, we formalize continuity through convergence under temporal refinement: when the same underlying signal is sampled more finely, the model's discrete predictions should approach a common continuous trajectory. The speed of this convergence separates models in a way that their continuous-time motivation alone does not. S4 and S5 exhibit stable continuous behavior, whereas the convergence of S6 is much more sensitive to input amplitude and selective dynamics.
- **What is a continuous task?** Sequential data have different degrees of continuous temporal structure, depending on how nearby observations relate to each other and how that relationship changes across time. Task continuity is not a binary label: we treat task continuity as a measurable property of the data rather than as a semantic label such as "language" or "time series." We introduce a metric that quantifies how strongly nearby elements in a sequence resemble one another relative to far-apart elements, placing datasets on a continuum from more discrete to more continuous.
- **How do model and task continuity interact?** Once both notions are defined, we can ask whether they align. Across controlled dynamical systems and Long-Range Arena (LRA) benchmarks, we

find that models with stronger continuity perform better as tasks become more continuous. This identifies continuity as an inductive bias for reasoning about architecture–task fit.

These notions treat continuity as an explanatory principle to help characterize models, measure tasks, and predict when the two are well matched. However, continuity can also be used constructively. As a concrete example, we revisit the state-space structure of SSMs. Prior work on efficient SSMs has focused on reducing the state dimension, thereby compressing the hidden dynamics [42, 7, 24]. We take a complementary view: *if a model behaves continuously in time, then one can also compress the time axis*. This leads to a stage-wise training strategy that begins with temporally subsampled inputs and progressively refines the resolution. While this is not the main focus of the paper, it illustrates a broader point: continuity is not only a diagnostic property, but a structure that can be exploited.

See Appendix A for related work. We summarize the main contributions of this paper as follows.

1. We introduce a theoretical framework for assessing continuity in SSM-like models via temporal refinement. This framework shows that a continuous-time origin does not by itself guarantee continuous behavior after discretization: S4 and S5 remain well behaved under refinement, whereas S6 can be considerably more affected by input magnitude and selective dynamics. We support our conclusion with numerical studies and experiments on 4 classes of dynamical systems.
2. We propose a data-driven metric for task continuity based on local temporal similarity. By comparing nearby elements in a sequence to far-apart ones, the metric places datasets along a spectrum from discrete to continuous. On both controlled dynamical systems and Long-Range Arena benchmarks [51], the resulting scores agree well with the intuitive continuity of the tasks.
3. We empirically demonstrate a strong match between model continuity and task continuity. Across S4, S6, and Transformer architectures, we observe that models with stronger temporal continuity become increasingly favorable as the task itself becomes more continuous. These results support continuity as a useful inductive bias for understanding model–task compatibility.
4. We show that continuity is useful not only for analysis, but also for algorithm design. Motivated by the temporal stability of S4, we invent a stage-wise training method that starts from temporally coarsened inputs and gradually restores full resolution. On five of the six LRA tasks, we achieve *any* reachable accuracy in less wall-clock time, and in many cases even improve the final accuracy.

2 A continuous framework for state-space models

Suppose $\mathbf{u}_0, \dots, \mathbf{u}_{L-1} \in \mathbb{R}^d$ is a sequence of input vectors. A sequential module maps this input to an output sequence $\mathbf{y}_0, \dots, \mathbf{y}_{L-1} \in \mathbb{R}^d$. For example, a (single-head) attention layer computes

$$\mathbf{y}_k = \mathbf{W}_V \mathbf{U} \operatorname{softmax}\Big(\mathbf{u}_k^\top \mathbf{W}_Q{}^\top \mathbf{W}_K \mathbf{U}/\sqrt{d}\Big)^\top, \qquad \mathbf{U} = [\mathbf{u}_0 \quad \cdots \quad \mathbf{u}_{L-1}] \in \mathbb{R}^{d\times L},$$

where we use gold to highlight the trainable parameters. The output is a weighted combination of $\mathbf{W}_V\mathbf{u}_0, \dots, \mathbf{W}_V\mathbf{u}_{L-1}$, and the entire computation is inherently discrete. For simplicity of exposition, we assume $d = 1$ from now on; extending the discussion to higher dimensions is straightforward. SSMs take a different route. They begin with a continuous-time dynamical system:

$$\mathbf{x}'(t) = \Delta(u(t))\big(\mathbf{A}\mathbf{x}(t) + \mathbf{B}(u(t))u(t)\big), \quad y(t) = \mathbf{C}(u(t))\mathbf{x}(t) + Du(t), \qquad \mathbf{x}(0) = \mathbf{0}. \tag{1}$$

Here $\mathbf{A} \in \mathbb{R}^{n\times n}$, $\mathbf{B} : \mathbb{R} \to \mathbb{R}^{n\times 1}$, $\mathbf{C} : \mathbb{R} \to \mathbb{R}^{1\times n}$, $D \in \mathbb{R}$, and $\Delta : \mathbb{R} \to (0, \infty)$. To connect this continuous system to discrete data, we view $u_0, \dots, u_{L-1}$ as samples of an underlying continuous signal $u(t)$. The output sequence is then obtained by discretizing the dynamics, for instance using zero-order hold (ZOH) or bilinear discretization:

$$\begin{aligned}
&\textbf{(ZOH)} \qquad \overline{\mathbf{A}}_k = \exp(\Delta(u_k)\mathbf{A}), \quad \overline{\mathbf{B}}_k = \mathbf{A}^{-1}(\overline{\mathbf{A}}_k - \mathbf{I}_n)\mathbf{B}(u_k), \quad \overline{\mathbf{C}}_k = \mathbf{C}(u_k),\\
&\textbf{(Bilinear)} \qquad \begin{aligned}
\overline{\mathbf{A}}_k &= \left(\mathbf{I}_n - (1/2)\Delta(u_k)\mathbf{A}\right)^{-1}\left(\mathbf{I}_n + (1/2)\Delta(u_k)\mathbf{A}\right),\\
\overline{\mathbf{B}}_k &= \left(\mathbf{I}_n - (1/2)\Delta(u_k)\mathbf{A}\right)^{-1}\Delta(u_k)\mathbf{B}(u_k), \quad \overline{\mathbf{C}}_k = \mathbf{C}(u_k),
\end{aligned}
\end{aligned} \tag{2}$$

followed by the discrete recursion

$$\mathbf{x}_{k+1} = \overline{\mathbf{A}}_k\mathbf{x}_k + \overline{\mathbf{B}}_k u_k, \quad y_k = \overline{\mathbf{C}}_k\mathbf{x}_k + Du_k, \qquad \mathbf{x}_0 = \mathbf{0}. \tag{3}$$

The key difference between S4 and S6 lies in how $\mathbf{B}$, $\mathbf{C}$, and Δ are defined:

$$\begin{aligned}
&\textbf{(S4)} \qquad \mathbf{B}(u) \equiv \mathbf{B}, \quad \mathbf{C}(u) \equiv \mathbf{C}, \quad \Delta(u) \equiv \Delta,\\
&\textbf{(S6)} \qquad \mathbf{B}(u) = \mathbf{B}u, \quad \mathbf{C}(u) = \mathbf{C}u, \quad \Delta(u) = \operatorname{softplus}(w_\Delta u + b_\Delta).
\end{aligned} \tag{4}$$

In S4, they are constant, resulting in a linear time-invariant system. In contrast, S6 introduces input dependency, leading to a nonlinear, time-varying system.

3 How continuous is a continuous model?

SSMs are often described as continuous-time models, but how accurate is this description in practice? To answer this, we need a concrete notion of continuity. A natural starting point is the following: the discrete outputs $y_0, \dots, y_{L-1}$ should closely match the continuous trajectory $y(t)$ evaluated at the same timestamps. However, this raises an immediate issue: in practice, we are only given a discrete sequence $u_0, \dots, u_{L-1}$, with no guarantee that it arises from a meaningful continuous signal $u(t)$. Without such a signal, the notion of a "continuous output" $y(t)$ is ill-defined.

To make progress, we adopt a slightly different viewpoint (see Figure 2). For now, we assume access to an underlying continuous input $u : [0, 1] \to \mathbb{R}^d$, which induces a continuous output $y(t)$. We then sample u on a uniform grid with step size $\tau > 0$:

$$u(0), u(\tau), \dots, u(\lfloor 1/\tau \rfloor \tau).$$

This produces a discrete input sequence, from which the model generates outputs.

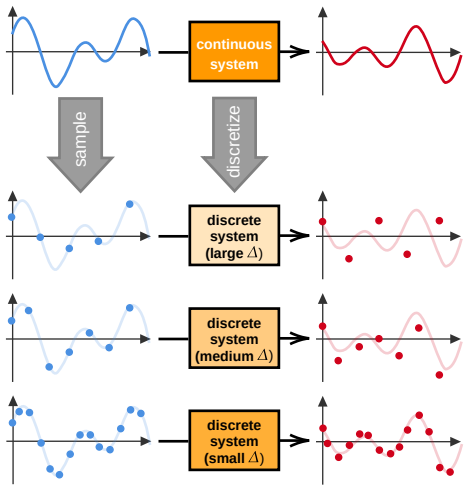


Figure 2: We assess a model's continuity by measuring how quickly the discrete output converges to the continuous one under temporal refinement.

We say that an SSM behaves continuously if, under temporal refinement, i.e., as $\tau \to 0^+$, the discrete outputs converge to the continuous trajectory at a fast rate. In other words, finer sampling of the same underlying signal should lead to more accurate approximations of the continuous solution. To be precise, at any fixed resolution the model and its predictions are, of course, discrete; "continuous" is our shorthand for this convergence property. There is only one subtlety that remains: the discretization rules in eq. (2) implicitly assume a unit time step and do not account for the refinement parameter τ. To properly capture the effect of temporal refinement, we modify the discretization as follows:

$$\begin{aligned}
&\textbf{(ZOH)} \quad \overline{\mathbf{A}}_k^\tau = \exp(\tau\Delta(u(\tau k))\mathbf{A}), \quad \overline{\mathbf{B}}_k^\tau = \mathbf{A}^{-1}(\overline{\mathbf{A}}_k^\tau - \mathbf{I}_n)\mathbf{B}(u(\tau k)), \quad \overline{\mathbf{C}}_k^\tau = \mathbf{C}(u(\tau k)), \\
&\textbf{(Bilinear)} \quad \overline{\mathbf{A}}_k^\tau = \left(\mathbf{I}_n - \frac{\tau\Delta(u(\tau k))}{2}\mathbf{A}\right)^{-1}\left(\mathbf{I}_n + \frac{\tau\Delta(u(\tau k))}{2}\mathbf{A}\right), \\
&\qquad\qquad \overline{\mathbf{B}}_k^\tau = \left(\mathbf{I}_n - \frac{\tau\Delta(u(\tau k))}{2}\mathbf{A}\right)^{-1}\tau\Delta(u(\tau k))\mathbf{B}(u(\tau k)), \quad \overline{\mathbf{C}}_k^\tau = \mathbf{C}(u(\tau k)).
\end{aligned} \tag{5}$$

The only change is to replace $u(k)$ with $u(\tau k)$ and Δ with $\tau\Delta$. Using this notion of continuity, we theoretically quantify the convergence rate of the continuous system in the general form of eq. (1).

Lemma 1. Let $u(t) : [0, 1] \to \mathbb{R}$ be L_u-Lipschitz continuous. Let $\mathbf{A} \in \mathbb{R}^{n\times n}$, $\mathbf{B} : \mathbb{R} \to \mathbb{R}^{n\times 1}$, $\mathbf{C} : \mathbb{R} \to \mathbb{R}^{1\times n}$, $D \in \mathbb{R}$, and $\Delta : \mathbb{R} \to (0, \infty)$ be given, where $\mathbf{B}$, $\mathbf{C}$, and Δ are L_B-, L_C-, and L_Δ-Lipschitz continuous (in $\|\cdot\|_2$-norm), respectively. Let

$$M_u = \sup_{t\in[0,1]} \|u(t)\|_2, \quad M_B = \sup_{t\in[0,1]} \|\mathbf{B}(u(t))\|_2, \quad M_C = \sup_{t\in[0,1]} \|\mathbf{C}(u(t))\|_2, \quad M_\Delta = \sup_{t\in[0,1]} \Delta(u(t))$$

be the maximum moduli of $\mathbf{B}$, $\mathbf{C}$, and Δ achieved by $u(t)$, respectively. Consider the output $y(t) : [0, 1] \to \mathbb{R}$ defined by the continuous system in eq. (1). For any $0 < \tau < 1$, let $\mathbf{y}^\tau = (y_0^\tau, \dots, y_{\lfloor 1/\tau \rfloor}^\tau)$ be the output of the discrete system

$$\mathbf{x}_{k+1}^\tau = \overline{\mathbf{A}}_k^\tau \mathbf{x}_k^\tau + \overline{\mathbf{B}}_k^\tau u(\tau k), \quad y_k^\tau = \overline{\mathbf{C}}_k^\tau \mathbf{x}_k^\tau + \overline{D}u(\tau k), \qquad \mathbf{x}_0^\tau = \mathbf{0},$$

where $\overline{\mathbf{A}}_k^\tau, \overline{\mathbf{B}}_k^\tau$, and $\overline{\mathbf{C}}_k^\tau$ are defined by either the ZOH or the bilinear formulas in eq. (5). Then, for every $1 \le k \le \lfloor 1/\tau \rfloor$, we have

$$|y_k^\tau - y(\tau k)| \le M_C \frac{M_\Delta(L_B M_u + M_B)L_u + L_\Delta L_u M_B M_u e^{M_\Delta \|\mathbf{A}\|_2}}{M_\Delta \|\mathbf{A}\|_2} (\exp(M_\Delta \|\mathbf{A}\|_2) - 1)\tau + \mathcal{O}(\tau^2).$$

Moreover, if $\overline{\mathbf{A}}_k^\tau, \overline{\mathbf{B}}_k^\tau$, and $\overline{\mathbf{C}}_k^\tau$ are obtained via the ZOH formula, then the $\mathcal{O}(\tau^2)$ term vanishes.

See Appendix B for a proof of the lemma and the corollaries presented in the rest of this section. Lemma 1 shows that the discrete output converges to the continuous output at a linear rate $\mathcal{O}(\tau)$. This is consistent with the classical behavior of standard discretizations such as ZOH and bilinear under smooth dynamics [40]. What we care about, however, is the constant in the $\mathcal{O}$-notation for this linear convergence rate. The following two corollaries are immediate consequences of Lemma 1.

Corollary 1. Suppose $\mathbf{B}, \mathbf{C}, D$, and Δ are defined by an S4 system (see eq. (4)). Then, under the setting of Lemma 1, we have

$$|y_k^\tau - y(\tau k)| \le \frac{\|\mathbf{C}\|_2 \|\mathbf{B}\|_2 L_u (\exp(\Delta \|\mathbf{A}\|_2) - 1)}{\|\mathbf{A}\|_2} \tau + \mathcal{O}(\tau^2).$$

Corollary 2. Suppose $\mathbf{B}, \mathbf{C}, D$, and Δ are defined by an S6 system (see eq. (4)). Then, under the setting of Lemma 1, we have

$$|y_k^\tau - y(\tau k)| \le \frac{\|\mathbf{W_C}\|_2 \|\mathbf{W_B}\|_2 M_u^2 L_u \left(2M_\Delta + |w_\Delta| M_u e^{M_\Delta \|\mathbf{A}\|_2}\right)}{M_\Delta \|\mathbf{A}\|_2} \left(e^{M_\Delta \|\mathbf{A}\|_2} - 1\right) \tau + \mathcal{O}(\tau^2),$$

where

$$M_\Delta = \text{softplus}(|w_\Delta| M_u + b_\Delta).$$

Corollary 1 and 2 show that both S4 and S6 converge at first order in the refinement scale τ, but the key difference lies in the prefactor. For S4, the continuity constant depends only on the fixed system parameters $(\mathbf{A}, \mathbf{B}, \mathbf{C}, \Delta)$ and the input regularity L_u. In particular, once the system is fixed, the dependence on the input enters only through its temporal smoothness. By contrast, for S6, the constant is substantially more sensitive: in addition to L_u, it depends on the input amplitude M_u, on the magnitudes of the projections $\|\mathbf{W_B}\|_2$ and $\|\mathbf{W_C}\|_2$, and on the selectivity parameter $|w_\Delta|$. Hence, the continuity breaks down for large-amplitude inputs or strongly selective delta-dynamics. Shrinking the input does not rescue S6 either: its output shrinks together with the error, so the relative error stays flat as the amplitude decreases and only grows as it increases (see Appendix F.1).

To illustrate these phenomena numerically, we sample 20 random S4 and S6 systems, together with 20 random input functions. For each system–input pair, we discretize the input using a range of step sizes $\tau \in \{2^{-10}, \dots, 2^{-2}\}$ and compare the resulting outputs $\mathbf{y}^\tau$ to the corresponding continuous trajectory $y(\cdot)$. The latter is approximated using a much finer discretization ($\tau = 2^{-14}$) together with a high-accuracy Runge–Kutta solver. We further probe the sensitivity to input amplitude by scaling each input function by a factor greater than 1 and repeating the experiment. While this scaling also increases the Lipschitz constant L_u, we report the relative error $\max_{1 \le k \le \lfloor 1/\tau \rfloor} |y_k^\tau - y(\tau k)| / \max_{1 \le k \le \lfloor 1/\tau \rfloor} |y(\tau k)|$ to normalize for this effect. As shown in Figure 3, the relative error of S4 remains stable under input scaling, whereas the error of S6 increases rapidly. This empirical behavior aligns with our theoretical predictions and highlights the fundamentally different continuity properties of the two models.

Empirical evidence for S6's instability under refinement. Our comparison of S4 and S6 so far is primarily theoretical, or at best numerical. You may wonder: do these findings actually matter in practice? To answer this, we study a downstream task based on dynamical systems data. In this setting, each sample is generated from an underlying parameterized dynamical system, and the goal is to recover its parameters from observed trajectories. To cover a broad range of behaviors, we consider four representative types of systems: a linear system (damped harmonic oscillator), a nonlinear system (Van der Pol oscillator), a stochastic system (Ornstein–Uhlenbeck process), and a chaotic system (forced Duffing oscillator). See Appendix C for more details.

We construct the training data by sampling these systems at a fixed rate $\tau_0 = 10^{-2}$ and, during training, adopt the ZOH discretization. Our emphasis here is not that S4 generally outperforms S6; instead, we compare a learned model against itself under two types of distribution shifts:

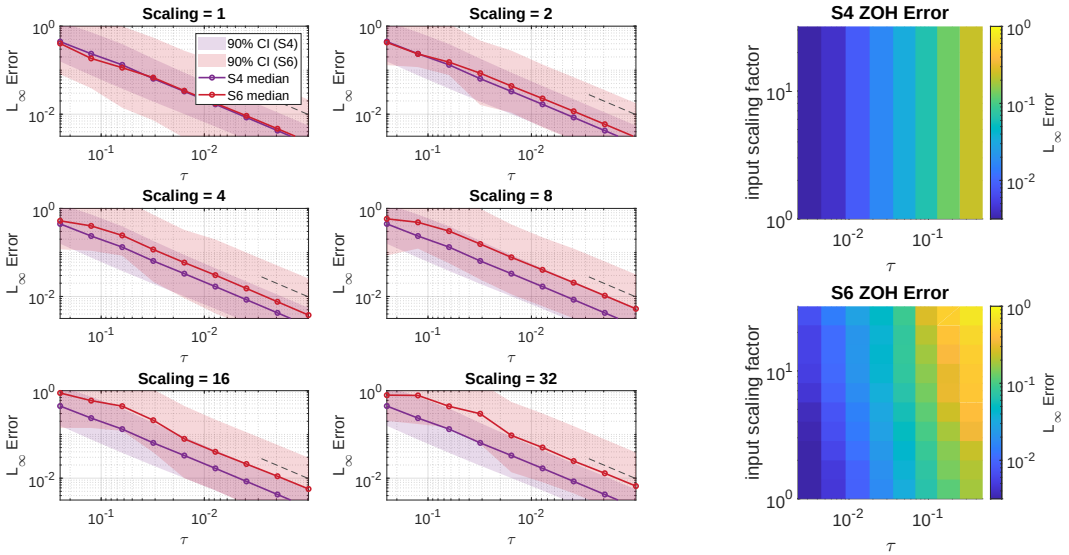


Figure 3: **Left:** Relative maximum error $\max_{1\le k\le \lfloor 1/\tau \rfloor} |y_k^\tau - y(\tau k)| / \max_{1\le k\le \lfloor 1/\tau \rfloor} |y(\tau k)|$ as $\tau \to 0^+$, evaluated over 20 randomly sampled system–input pairs. The input is scaled by factors ranging from 1 to 32. The gray reference line indicates first-order (linear) convergence. **Right:** Heatmap of the same relative error. For S4, the error remains stable as the input is scaled, whereas for S6, it increases significantly with input magnitude.

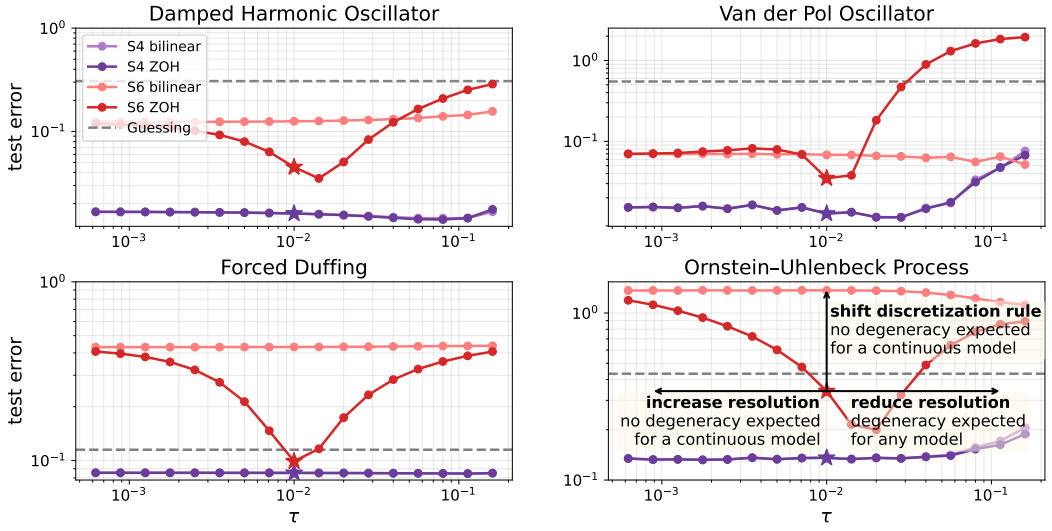


Figure 4: Test error as a function of the inference-time sampling rate τ across four dynamical systems. For each model, we evaluate both ZOH and bilinear discretizations. The star marks the resolution $\tau_0 = 10^{-2}$ and the ZOH discretization used for training, and the dashed line indicates the expected error of a guessing baseline. S4 is stable across changes in τ and discretization, but S6 is sensitive to both, with performance degrading under refinement and when switching discretization schemes. (The S4 bilinear curve is hidden underneath the S4 ZOH curve.)

1. **Changing resolution.** Still using ZOH discretization, we resample the same systems at a different rate τ at inference time, and adjust the Δ parameter in the S4 or S6 model by a factor of τ/τ_0 to account for the change in resolution. Intuitively, if a model has learned the underlying continuous dynamics, then refining the sampling (i.e., $\tau < \tau_0$) should not degrade its performance (see also [46]). As shown in Figure 4, this holds for S4: its performance remains stable under temporal

refinement. In contrast, S6 degrades as the resolution increases, indicating that its discrete outputs do not faithfully track the underlying continuous dynamics.

2. **Changing discretization rule.** We also vary the discretization method at inference time, switching from ZOH to bilinear. This provides a complementary test: if a model truly captures the continuous dynamics, its predictions should be largely invariant to the choice of discretization. As shown in Figure 4, S4 is robust to this change, with little difference between the two schemes. In contrast, S6 exhibits noticeable degradation when switching to bilinear discretization, even at the same sampling rate $\tau = \tau_0$ used for training.

4 Are continuous models better at learning continuous tasks?

So far, we have shown that certain models (e.g., S4) more faithfully capture underlying continuous dynamics than others (e.g., S6). Intuitively, such models should be better suited for continuous tasks. But does this intuition actually hold in practice? In this section, we first design a controlled experiment to test this hypothesis, and then move toward formalizing the notion of continuous tasks so that we can examine this effect across a broader range of problems. Throughout, we focus on three representative model classes: S4 (continuous formulation and behavior), S6 (continuous formulation but not in behavior), and Transformers (purely discrete formulation).

To begin, we construct a controlled setting in which we can gradually vary the "degree of continuity" of a task. At first glance, this might seem straightforward: one could simply compare performance across tasks with different levels of continuity. However, this approach introduces two key challenges:

1. **Control over difficulty.** Tasks differ in intrinsic difficulty. That a model is better at a "continuous" task than a "discrete" one does not necessarily indicate an advantage in handling continuity.
2. **Control over other inductive biases.** Models and tasks may also differ along other dimensions, such as frequency content or long-range dependency. Without careful control, other inductive biases of models can confound any conclusions about continuity.

To address the first issue, we compare the *relative performance* of S4, S6, and Transformers as the task transitions from discrete to continuous. To address the second, we base all experiments on the same underlying dynamical systems, thereby isolating continuity as the primary varying factor. Concretely, we consider a forecasting problem on trajectories generated from the four dynamical systems described in section 3. Given a normalized trajectory $u : [0, 1] \to [-1, 1]$, we construct two representations in $\mathbb{R}^{16}$:

$$\mathbf{U}_{\text{cts}}(t) = u(t)\mathbf{v}, \qquad \mathbf{U}_{\text{dct}}(t) = \sum_{i=1}^{16} \mathbb{1}_{\{u(t)\in R_i\}}\mathbf{q}_i,$$

where $\mathbf{v} \in \mathbb{R}^{16}$ is a fixed random unit vector, $\{\mathbf{q}_i\}_{i=1}^{16}$ form an orthonormal basis in $\mathbb{R}^{16}$, and $\{R_i\}_{i=1}^{16}$ partition $[-1, 1]$ into equal intervals. The first embedding preserves the continuity of the signal, while the second introduces quantization and produces a more discrete representation (see [60] for more discussion). We then form models' input by interpolating between these two regimes by defining

$$\mathbf{U}_\eta(t) = \eta\mathbf{U}_{\text{cts}}(t) + (1 - \eta)\mathbf{U}_{\text{dct}}(t), \qquad \eta \in [0, 1]. \tag{6}$$

Therefore, as η increases from 0 to 1, the task transitions smoothly from discrete to continuous.

For each fixed continuity level η, we train an S4, an S6, and a Transformer to forecast the dynamical systems. Results in Figure 5 (Left) reveal a clear pattern: on more discrete tasks (i.e., when η is small), S4 underperforms S6 and Transformers, but as the task becomes more continuous, S4 begins to outperform both. The ordering between S6 and Transformers themselves is less telling: it shifts with how sharply attention selects tokens and how strongly Δ depends on the input, whereas the crossover with S4 survives all of these choices (see Appendix F.3). Importantly, all tasks are derived from the same underlying dynamical systems; only the representation changes. This makes clear that continuity, by itself, can substantially alter how these models perform. We present results for a wider range of model classes in Table 2.

While the controlled experiment reveals the advantage of continuous models on continuous tasks (and vice versa), most real-world problems do not admit a representation such as eq. (6). To study continuity in a broader setting, we introduce a metric that can be applied to a general sequential dataset. At a high level, we measure how similar nearby elements in a sequence are, compared to

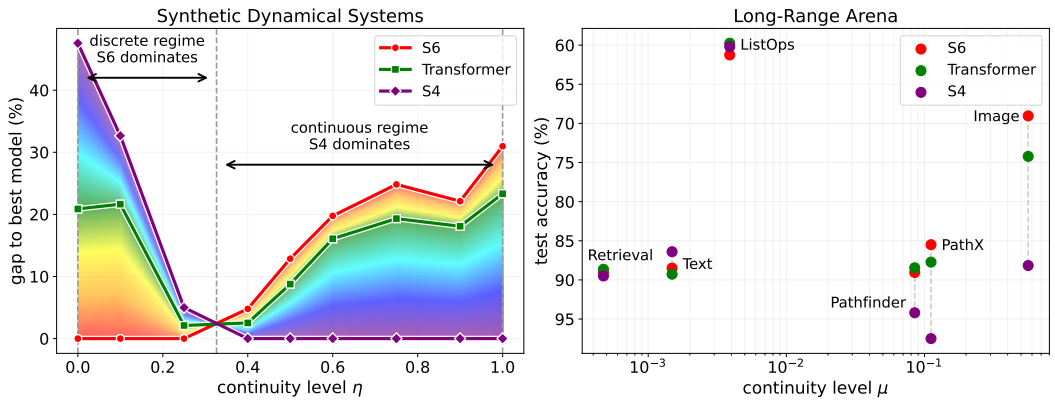


Figure 5: **Left:** Relative L_∞ forecasting error gap to the best model, defined as $\mathrm{gap}(\mathcal{M}) = (e_\mathcal{M} - e_{\min})/e_{\min}$, where $e_{\min} = \min_{\mathcal{M}'} e_{\mathcal{M}'}$ and $e_\mathcal{M}$ is the L_∞ forecasting error of model $\mathcal{M}$, for S4, S6, and Transformers on dynamical systems data as the underlying representation becomes more continuous. **Right:** Computed continuity score $\mu = \sum_{t=1}^{16} \mu_t/16$ for the six LRA tasks, together with the reported accuracies of S4, S6, and Transformers. In both panels, S4 gradually overtakes S6 and Transformers as the tasks become more continuous. See Appendix D for more details.

pairs that are far apart. A dataset with stronger continuity should exhibit higher local similarity that decays gradually with temporal separation. To make this precise, let $\mathbf{u} = (\mathbf{u}_0, \ldots, \mathbf{u}_{L-1})$ denote a sequence drawn from the dataset distribution $\mathcal{D}$, where each $\mathbf{u}_k \in \mathbb{R}^d$. Let $K : \mathbb{R}^d \times \mathbb{R}^d \to [0, 1]$ be a similarity kernel, e.g., the cosine kernel $K(\mathbf{x}, \mathbf{y}) = \left(1 + \mathbf{x}^\top \mathbf{y}/\|\mathbf{x}\|_2 \, \|\mathbf{y}\|_2\right)/2$. For a lag parameter $t \geq 1$ and a fixed gap $\varrho \gg t$, we define

$$\mu_t := \frac{\mathbb{E}_{\mathbf{u}\sim\mathcal{D}}\mathbb{E}_{k\sim\mathrm{Unif}(0:L-t-1)}\left[K(\mathbf{u}_k, \mathbf{u}_{k+t})\right] - \beta}{\mathbb{E}_{\mathbf{u}\sim\mathcal{D}}\mathbb{E}_{k\sim\mathrm{Unif}(0:L-1)}\left[K(\mathbf{u}_k, \mathbf{u}_k)\right] - \beta}, \quad \beta = \mathbb{E}_{\mathbf{u}\sim\mathcal{D}}\mathbb{E}_{(k,k')\sim\mathrm{Unif}(\{|k-k'|>\varrho\})}\left[K(\mathbf{u}_k, \mathbf{u}_{k'})\right].$$

In practice, we approximate these expectations by averaging over sequences in a dataset, uniformly sampled valid positions k, and randomly sampled far-apart pairs (k, k'). For an overall continuity score, we aggregate the local metrics over a range of small lags, for example, by taking $\mu = \sum_{t=1}^{T} w_t \mu_t$, where the weights w_t decrease with t, and T sets the maximum lag under consideration.

We do not wish to claim that this is the best metric for measuring dataset continuity, and we hope future work will develop more alternatives and compare them systematically. In particular, smoothness, locality, and low-frequency structure all show up as local similarity, so μ_t is best read as a practical proxy for continuity rather than a quantity that isolates it from these close relatives. Still, the metric μ_t proposed here has several appealing properties: it does not depend explicitly on the sequence length L; its dependence on the ambient dimension d is entirely absorbed into the choice of kernel K, so once K is specified, the metric applies seamlessly across different representation spaces; the correction term β removes the background similarity between unrelated or far-apart elements, so that μ_t reflects local continuity rather than the global geometry of the representation space (e.g., see [61]); and unlike a pointwise quantity such as a Lipschitz constant, μ_t provides a more stable measurement by aggregating the continuity structure across the entire sequences.

By computing μ for the six Long-Range Arena (LRA) tasks, we indeed find that tasks that are intuitively more continuous, such as flattened images, receive much higher continuity scores than intuitively discrete tasks such as text[2]; moreover, μ indeed positively correlates to η in our synthetic dataset (see Figure 8). Using this metric, in Figure 5 (Right), we observe that S4 gradually outperforms S6 and Transformers as the tasks become more continuous. This provides further evidence that models with stronger continuity are better suited to the more continuous tasks. A controlled intervention points the same way: weakening the input dependence of Δ in S6 makes it more continuous and lifts its accuracy on Image, Pathfinder, and PathX, while leaving the more discrete tasks roughly unchanged (see Table 1 and Appendix F.4).

[2]To compute μ, one first needs to embed the raw data into a hidden space. Here, we use the encoders of pretrained S4 models; using the encoders of S6 models or Transformers leads to very similar scores, and so do other kernels and lag ranges (see Appendix F.2).

Table 1: Test accuracy (in %) on the six LRA tasks of S6 models whose step size is $\Delta(u) = \text{softplus}(\alpha w_\Delta u + b_\Delta)$, for the standard choice $\alpha = 1$, a weakened input dependence $\alpha = 0.1$, and no input dependence $\alpha = 0$. The last row reports the relative discretization error of the same models in the setting of Figure 3 with an input scaling factor of 32; S4 attains 0.0025 in this setting.

	S6 ($\alpha = 1$)	S6 ($\alpha = 0.1$)	S6 ($\alpha = 0$)
ListOps	61.25	60.30	58.50
Retrieval	89.03	88.50	88.17
Text	88.49	88.30	87.94
Image	69.04	73.19	82.38
Pathfinder	89.05	88.68	90.53
PathX	85.50	88.08	89.12
Relative discretization error (scale 32)	0.0103	0.0054	0.0026

5 Beyond inductive biases: What does continuity buy us?

At this point, we have identified continuity as an important inductive bias in sequential modeling. This has several consequences. For instance, it can help us anticipate which models are more likely to succeed on continuous or discrete tasks without fully training them, and it suggests how one might tune SSMs toward problems with different levels of continuity. But is continuity only useful as an inductive bias? In fact, continuity has broader algorithmic implications. We conclude the paper with one such application: accelerating training.

Input: Training set $\mathcal{D} = \{(\mathbf{u}^{(i)}, y^{(i)})\}$, stage strides $r_1 > \cdots > r_S = 1$, epochs $E_1, \ldots, E_S$
Output: Trained model parameters
Initialize S4 parameters $(\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{D}, \Delta)$
$r_0 \leftarrow r_1$
for $s = 1, \ldots, S$ **do**
  $\Delta \leftarrow \Delta \cdot (r_s / r_{s-1})$ ▷ See eq. (5)
  for all $(\mathbf{u}^{(i)}, y^{(i)}) \in \mathcal{D}$ **do**
    Form subsampled input $\mathbf{u}^{(i,s)} = \left(\mathbf{u}_0^{(i)}, \mathbf{u}_{r_s}^{(i)}, \mathbf{u}_{2r_s}^{(i)}, \ldots\right)^a$
  end for
  Train the model on dataset $\mathcal{D}^{(s)} = \{(\mathbf{u}^{(i,s)}, y^{(i)})\}$ for E_s epochs.
end for

[a] Average pooling can be used instead for discrete tasks.

Algorithm 1: Stage-wise training via temporal subsampling. At each stage, the input is subsampled by stride r_s and Δ is scaled by the same factor.

Training is often the most computationally expensive part of using sequential models. Existing acceleration methods, often inspired by reduced-order modeling, typically focus on compressing the state dimension, i.e., the hidden state $\mathbf{x}$ in eq. (1) [7]. Such approaches can reduce the cost of later training epochs. Here, we show that continuity in S4 suggests a complementary direction: as observed in Figure 4, S4 remains stable under temporal refinement. This motivates a natural speculation: if an S4 model can adapt well to higher-resolution data, why not begin training at a lower resolution?

This leads to a stage-wise training strategy. We start by subsampling the input sequence to a coarse temporal resolution, train the model on this cheaper version of the task, and then progressively refine the resolution while continuing to train the same model (see Algorithm 1). One may also view each stage as fine-tuning the model on a more resolved version of the data.

Conceptually, our method operates entirely on the time axis, so it can be combined with state-pruning techniques for additional speedups. Compared to those methods, it has two notable differences:

1. Our method lowers the cost of the early training stages by initially using subsampled sequences, so computation is cheap before the model is exposed to the full-resolution data. As a result, it can reach a given accuracy substantially earlier in wall-clock time.
2. Our method does not modify or compress the model itself, so it preserves the full expressive power of the original architecture. The tradeoff is that it does not reduce inference cost.

Our method is also connected to the multi-stage framework of [8], which progressively increases network depth during training. The two approaches nonetheless rely on different notions of continuity: [8] exploits continuity in model depth, whereas we exploit continuity in time for continuous sequential models. In Figure 6, we show that, for five of the six LRA tasks, the time–accuracy curve of our method consistently stays above that of standard training (see Appendix E for more

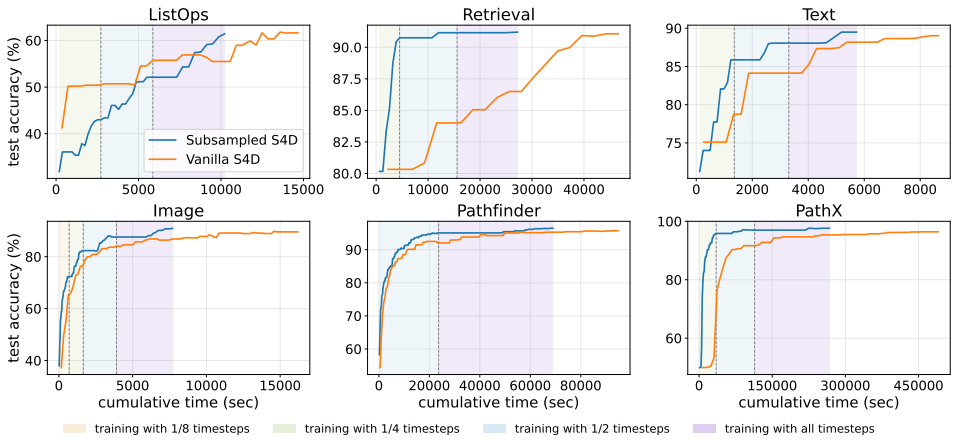


Figure 6: Test accuracy plotted against cumulative training time on the six LRA tasks. We compare standard training with our stage-wise subsampling strategy, which starts from low-resolution inputs and progressively refines the temporal resolution during training. On five of the six tasks, the time–accuracy curve of our method stays above that of standard training, indicating that it reaches essentially any target accuracy in less wall-clock time. On several tasks, especially the more continuous ones, it also attains a higher final accuracy. All experiments are done on a single NVIDIA L40S GPU.

details). In other words, our approach reaches *any* attainable accuracy in less time. Moreover, on several tasks — especially the more continuous ones, such as Pathfinder, PathX, and Image — it also achieves a higher final accuracy than standard training. One possible explanation is that, for tasks with stronger continuous structure, starting from low-resolution data encourages the model to learn coarse patterns first before fitting finer-scale details at later stages. This may act as a form of implicit regularization. We leave a more precise understanding of this phenomenon to future work. In addition, our subsampling method echoes the significance of model continuity; in particular, it does not help the more discrete-in-practice models such as S6 (see Figure 10). Final accuracies averaged over three random seeds are reported in Appendix F.5. The same stability of S4 also shows up on real-world forecasting data when observations go missing at test time (see Appendix F.6).

6 Conclusion

In this paper, we propose continuity as a principle for understanding when sequential models match the temporal structure of their tasks. Our results show that models differ in how their predictions behave under temporal refinement, tasks differ in how much continuous temporal structure their sequences contain, and performance improves when these two forms of continuity are aligned. This perspective separates continuous-time motivation from continuous behavior: although S4, S5, and S6 all arise from state-space formulations, they have different refinement behaviors once implemented as discrete models. More broadly, understanding task continuity together with the continuous inductive bias of an architecture can help predict which models should be favored in which temporal regimes. In practice, a continuous-time formulation alone should not decide model choice. A more useful habit is to ask whether the data change smoothly, whether the sampling rate may change, and whether observations may go missing, and then to test the model under temporal coarsening; our score can assist this diagnosis alongside validation performance. This suggests that continuity is a useful law-like organizing principle for sequential modeling. Future work should develop a broader theory of how continuity interacts with other inductive biases.

Acknowledgments and Disclosure of Funding

We would like to acknowledge support from the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research, EXPRESS: 2025 Exploratory Research for Extreme-Scale Science program, under Contract Number DE-AC02-05CH11231 at Berkeley Lab. The views, opinions, and/or findings expressed are those of the authors and should not be interpreted as representing the official views or policies of the Department of Defense or the U.S. Government.

References

[1] Carmen Amo Alonso, Jerome Sieber, and Melanie N Zeilinger. State space models as foundation models: A control theoretic overview. In *2025 American Control Conference (ACC)*, pages 146–153. IEEE, 2025.

[2] Ido Amos, Jonathan Berant, and Ankit Gupta. Never train from scratch: Fair comparison of long-sequence models requires data-driven priors. *arXiv preprint arXiv:2310.02980*, 2023.

[3] Abdul Fatir Ansari, Lorenzo Stella, Caner Turkmen, Xiyuan Zhang, Pedro Mercado, Huibin Shen, Oleksandr Shchur, Syama Sundar Rangapuram, Sebastian Pineda Arango, Shubham Kapoor, et al. Chronos: Learning the language of time series. *arXiv preprint arXiv:2403.07815*, 2024.

[4] Athanasios C Antoulas. *Approximation of large-scale dynamical systems*. SIAM, 2005.

[5] Iz Beltagy, Matthew E Peters, and Arman Cohan. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*, 2020.

[6] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[7] Makram Chahine, Philipp Nazari, Daniela Rus, and T Konstantin Rusch. The curious case of in-training compression of state space models. *arXiv preprint arXiv:2510.02823*, 2025.

[8] Bo Chang, Lili Meng, Eldad Haber, Frederick Tung, and David Begert. Multi-level residual networks from dynamical systems view. *arXiv preprint arXiv:1710.10348*, 2017.

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

[10] Junyoung Chung, Sungjin Ahn, and Yoshua Bengio. Hierarchical multiscale recurrent neural networks. *arXiv preprint arXiv:1609.01704*, 2016.

[11] Abhimanyu Das, Weihao Kong, Rajat Sen, and Yichen Zhou. A decoder-only foundation model for time-series forecasting. *arXiv preprint arXiv:2310.10688*, 2023.

[12] Edward De Brouwer, Jaak Simm, Adam Arany, and Yves Moreau. Gru-ode-bayes: Continuous modeling of sporadically-observed time series. *Advances in neural information processing systems*, 32, 2019.

[13] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*, pages 4171–4186, 2019.

[14] M Reza Ebrahimi, Michaël Defferrard, Sunny Panchal, and Roland Memisevic. On the" induction bias" in sequence models. *arXiv preprint arXiv:2602.18333*, 2026.

[15] Daniel Y Fu, Elliot L Epstein, Eric Nguyen, Armin W Thomas, Michael Zhang, Tri Dao, Atri Rudra, and Christopher Ré. Simple hardware-efficient long convolutions for sequence modeling. In *International Conference on Machine Learning*, pages 10373–10391. PMLR, 2023.

[16] Keith Glover. All optimal hankel-norm approximations of linear multivariable systems and their l, ∞-error bounds. *International journal of control*, 39(6):1115–1193, 1984.

[17] Lars Graf, Thomas Ortner, Stanisław Woźniak, and Angeliki Pantazi. Flowstate: Sampling-rate invariant time series foundation model with dynamic forecasting horizons. In *Recent Advances in Time Series Foundation Models Have We Reached the'BERT Moment'?*, 2025.

[18] Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*, 2023.

[19] Albert Gu, Karan Goel, Ankit Gupta, and Christopher Ré. On the parameterization and initialization of diagonal state space models. *Advances in neural information processing systems*, 35:35971–35983, 2022.

[20] Albert Gu, Karan Goel, and Christopher Ré. Efficiently modeling long sequences with structured state spaces. *arXiv preprint arXiv:2111.00396*, 2021.

[21] Albert Gu, Isys Johnson, Karan Goel, Khaled Saab, Tri Dao, Atri Rudra, and Christopher Ré. Combining recurrent, convolutional, and continuous-time models with linear state space layers. *Advances in neural information processing systems*, 34:572–585, 2021.

[22] Albert Gu, Isys Johnson, Aman Timalsina, Atri Rudra, and Christopher Ré. How to train your hippo: State space models with generalized orthogonal basis projections. *arXiv preprint arXiv:2206.12037*, 2022.

[23] Serkan Gugercin and Athanasios C Antoulas. A survey of model reduction by balanced truncation and some new results. *International Journal of Control*, 77(8):748–766, 2004.

[24] Minseon Gwak, Seongrok Moon, Joohwan Ko, and PooGyeon Park. Layer-adaptive state pruning for deep state space models. *Advances in Neural Information Processing Systems*, 37:10613–10645, 2024.

[25] 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.

[26] 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, pages 7657–7666, 2021.

[27] Kareem Hegazy, Michael W Mahoney, and N Benjamin Erichson. Recency biased causal attention for time-series forecasting. In *The 29th International Conference on Artificial Intelligence and Statistics*, 2026.

[28] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

[29] Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are rnns: Fast autoregressive transformers with linear attention. In *International conference on machine learning*, pages 5156–5165. PMLR, 2020.

[30] Suyong Kim, Weiqi Ji, Sili Deng, Yingbo Ma, and Christopher Rackauckas. Stiff neural ordinary differential equations. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 31(9), 2021.

[31] Aakash Lahoti, Kevin Y Li, Berlin Chen, Caitlin Wang, Aviv Bick, J Zico Kolter, Tri Dao, and Albert Gu. Mamba-3: Improved sequence modeling using state space principles. *arXiv preprint arXiv:2603.15569*, 2026.

[32] Peter D Lax and Robert D Richtmyer. Survey of the stability of linear finite difference equations. *Communications on pure and applied mathematics*, 9(2):267–293, 1956.

[33] Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Fourier neural operator for parametric partial differential equations. *arXiv preprint arXiv:2010.08895*, 2020.

[34] Zongyi Li, Nikola Kovachki, Kamyar Azizzadenesheli, Burigede Liu, Kaushik Bhattacharya, Andrew Stuart, and Anima Anandkumar. Neural operator: Graph kernel network for partial differential equations. *arXiv preprint arXiv:2003.03485*, 2020.

[35] Fusheng Liu and Qianxiao Li. Autocorrelation matters: Understanding the role of initialization schemes for state space models. *arXiv preprint arXiv:2411.19455*, 2024.

[36] Lu Lu, Pengzhan Jin, and George Em Karniadakis. Deeponet: Learning nonlinear operators for identifying differential equations based on the universal approximation theorem of operators. *arXiv preprint arXiv:1910.03193*, 2019.

[37] Martin Fluder. Mamba: Long range arena. `https://github.com/fluderm/MLRA`, 2024. GitHub repository. Accessed: 2026-04-30.

[38] Tom M Mitchell. The need for biases in learning generalizations. 1980.

[39] Nicola Muca Cirone, Antonio Orvieto, Benjamin Walker, Cristopher Salvi, and Terry Lyons. Theoretical foundations of deep selective state-space models. *Advances in Neural Information Processing Systems*, 37:127226–127272, 2024.

[40] Alan Oppenheim and Ronald Schafer. *Discrete-Time Signal Processing* . Pearson Deutschland, 2013.

[41] Antonio Orvieto, Samuel L Smith, Albert Gu, Anushan Fernando, Caglar Gulcehre, Razvan Pascanu, and Soham De. Resurrecting recurrent neural networks for long sequences. In *International conference on machine learning*, pages 26670–26698. PMLR, 2023.

[42] Apurba Prasad Padhy, Fernando Camacho, and Saibal Mukhopadhyay. Aire-prune: Asymptotic impulse-response energy for state pruning in state space models. *arXiv preprint arXiv:2602.00534*, 2026.

[43] Rom N Parnichkun, Stefano Massaroli, Alessandro Moro, Jimmy TH Smith, Ramin Hasani, Mathias Lechner, Qi An, Christopher Ré, Hajime Asama, Stefano Ermon, et al. State-free inference of state-space models: The transfer function approach. *arXiv preprint arXiv:2405.06147*, 2024.

[44] Xihao Piao, Zheng Chen, Taichi Murayama, Yasuko Matsubara, and Yasushi Sakurai. Fredformer: Frequency debiased transformer for time series forecasting. In *Proceedings of the 30th ACM SIGKDD conference on knowledge discovery and data mining*, pages 2400–2410, 2024.

[45] Michael Poli, Stefano Massaroli, Junyoung Park, Atsushi Yamashita, Hajime Asama, and Jinkyoo Park. Graph neural ordinary differential equations. *arXiv preprint arXiv:1911.07532*, 2019.

[46] Alejandro F Queiruga, N Benjamin Erichson, Dane Taylor, and Michael W Mahoney. Continuous-in-depth neural networks. *arXiv preprint arXiv:2008.02389*, 2020.

[47] Nasim Rahaman, Aristide Baratin, Devansh Arpit, Felix Draxler, Min Lin, Fred Hamprecht, Yoshua Bengio, and Aaron Courville. On the spectral bias of neural networks. In *International conference on machine learning*, pages 5301–5310. PMLR, 2019.

[48] 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.

[49] Jerome Sieber, Carmen A Alonso, Alexandre Didier, Melanie N Zeilinger, and Antonio Orvieto. Understanding the differences in foundation models: Attention, state space models, and recurrent neural networks. *Advances in Neural Information Processing Systems*, 37:134534–134566, 2024.

[50] Jimmy TH Smith, Andrew Warrington, and Scott W Linderman. Simplified state space layers for sequence modeling. *arXiv preprint arXiv:2208.04933*, 2022.

[51] Yi Tay, Mostafa Dehghani, Samira Abnar, Yikang Shen, Dara Bahri, Philip Pham, Jinfeng Rao, Liu Yang, Sebastian Ruder, and Donald Metzler. Long range arena: A benchmark for efficient transformers. *arXiv preprint arXiv:2011.04006*, 2020.

[52] Lloyd N Trefethen. *Approximation theory and approximation practice, extended edition*. SIAM, 2019.

[53] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[54] Shida Wang and Qianxiao Li. Stablessm: Alleviating the curse of memory in state-space models through stable reparameterization. *arXiv preprint arXiv:2311.14495*, 2023.

[55] Yihan Wang, Lujun Zhang, Annan Yu, N Benjamin Erichson, and Tiantian Yang. A deep state space model for rainfall-runoff simulations. *Water Resources Research*, 61(12):e2025WR039888, 2025.

[56] Xinyi Wu, Yifei Wang, Stefanie Jegelka, and Ali Jadbabaie. On the emergence of position bias in transformers. *arXiv preprint arXiv:2502.01951*, 2025.

[57] Songlin Yang, Bailin Wang, Yu Zhang, Yikang Shen, and Yoon Kim. Parallelizing linear transformers with the delta rule over sequence length. *Advances in neural information processing systems*, 37:115491–115522, 2024.

[58] Annan Yu and N Benjamin Erichson. Block-biased mamba for long-range sequence processing. *arXiv preprint arXiv:2505.09022*, 2025.

[59] Annan Yu, Dongwei Lyu, Soon Hoe Lim, Michael W Mahoney, and N Benjamin Erichson. Tuning frequency bias of state space models. *arXiv preprint arXiv:2410.02035*, 2024.

[60] Annan Yu, Danielle C Maddix, Boran Han, Xiyuan Zhang, Abdul Fatir Ansari, Oleksandr Shchur, Christos Faloutsos, Andrew Gordon Wilson, Michael W Mahoney, and Yuyang Wang. Understanding the implicit biases of design choices for time series foundation models. *arXiv preprint arXiv:2510.19236*, 2025.

[61] Annan Yu, Danielle C Maddix, Boran Han, Xiyuan Zhang, Abdul Fatir Ansari, Oleksandr Shchur, Christos Faloutsos, Andrew Gordon Wilson, Michael W Mahoney, and Yuyang Wang. Understanding transformers for time series: Rank structure, flow-of-ranks, and compressibility. *arXiv preprint arXiv:2510.03358*, 2025.

[62] Annan Yu, Michael W Mahoney, and N Benjamin Erichson. Hope for a robust parameterization of long-memory state space models. *arXiv preprint arXiv:2405.13975*, 2024.

[63] Annan Yu, Arnur Nigmetov, Dmitriy Morozov, Michael W Mahoney, and N Benjamin Erichson. Robustifying state-space models for long sequences via approximate diagonalization. *arXiv preprint arXiv:2310.01698*, 2023.

[64] Annan Yu and Alex Townsend. Leveraging the hankel norm approximation and data-driven algorithms in reduced order modeling. *Numerical Linear Algebra with Applications*, 31(4):e2555, 2024.

[65] Annan Yu, Yunan Yang, and Alex Townsend. Tuning frequency bias in neural network training with nonuniform data. *arXiv preprint arXiv:2205.14300*, 2022.

[66] Nikola Zubic, Mathias Gehrig, and Davide Scaramuzza. State space models for event cameras. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 5819–5828, 2024.

A Related Work

Continuous Models. Continuous-time neural models arose from the observation that many sequential and dynamical processes are more naturally described by latent states evolving on a continuous time axis than by fixed-step recursions. A major modern entry point was Neural ODEs [9, 30, 45], which replaced a finite stack of residual layers with a learned vector field integrated by an ODE solver, thereby turning depth itself into a continuous variable. This perspective quickly led to continuous-time latent-variable models for irregularly sampled data, such as Latent ODEs and ODE-RNNs [48], where hidden states evolve continuously between observations and are updated when new data arrive. A related line of work emphasizes *adaptive* or *gated* continuous dynamics: GRU-ODE-Bayes imports GRU-style gating into a continuous-time evolution rule [12], while Liquid Time-Constant networks and their closed-form variants modulate effective time constants and input responses through state- and input-dependent interactions [26, 25]. From this viewpoint, gating can be seen as a mechanism for making the underlying dynamics selective or input-adaptive, rather than purely time-invariant. Our work is conceptually adjacent to this literature, but asks a different question: when a model is presented as continuous-time, to what extent does its discrete behavior actually preserve that continuity in practice? A closely related perspective is developed in [46], which argues that the connection between deep networks and continuous dynamics should be understood through the lens of numerical integration more carefully, rather than through a purely formal Euler-style analogy, and emphasizes invariance across different discrete realizations of the same underlying dynamics.

Continuity of State-space Models. State-space models form another major class of continuous models in modern sequence learning: the original linear state-space layer and S4 explicitly start from a continuous-time system and then discretize it for efficient computation [20], while S4D [20], S5 [50], and S6 [18] simplify, generalize, or make selective this same underlying state-space viewpoint. In particular, S4 builds an efficient structured parameterization of a continuous-time linear system, S4D shows that a diagonal realization can retain much of this behavior when properly initialized, S5 recasts the layer as a multi-input multi-output SSM with efficient scans, and S6/Mamba makes the discretized dynamics input-dependent through selectivity.

This continuous perspective is not limited to the canonical S4/S4D/S5 family. Classical systems theory gives a tight correspondence between state-space realizations, transfer functions, and convolutional descriptions [4], and this viewpoint has become increasingly explicit in machine learning: [21] frames sequence layers as recurrent, convolutional, and continuous-time models simultaneously; [62] develops a different continuous-time parameterization through Hankel operators; and even seemingly more "discrete" sequence models such as linear recurrent units [41] and smooth long convolutions [15] can often be interpreted through equivalent linear dynamical or transfer-function descriptions [43, 63].

A growing line of work leverages this continuity directly rather than treating it as background intuition. For example, [22] gives a more explicit continuous interpretation of HiPPO-style initialization; [35] studies initialization through temporal structure; and [59] analyzes how continuous-time transfer functions induce frequency preferences and how those preferences can be modified. On the theoretical side, [54] studies long-term memory in SSMs from the viewpoint of parameterization, showing that stable reparameterizations can preserve expressive long-range dynamics while improving optimization stability, and [39] proposes a continuous-time-inspired framework for understanding selective SSMs via controlled dynamical systems and rough-path tools.

This continuity has also been used in applications, especially when the sampling rate may vary across training and deployment. In event-based vision, S4/S5-style models have been used precisely because their learnable time-scale parameter allows inference at frequencies different from those seen during training [66]; in time-series forecasting, [17] uses an S5-based encoder together with continuous-time scaling to target sampling-rate invariance; and in scientific machine learning, deep SSMs have recently been shown to outperform Transformers on rainfall–runoff simulation [55]. These works are complementary to ours: they exploit the continuous formulation of SSMs in practice, whereas our focus is to ask when that continuous interpretation is actually faithful at the level of the induced discrete model.

Inductive Biases of Sequential Models. Inductive bias is a foundational concept in machine learning: in the classical view, some bias is necessary for any learner to generalize beyond the training data, and modern deep learning can often be understood as the interaction between architectural,

optimization, and data-induced biases [38]. In general neural networks, one of the most studied examples is frequency bias or spectral bias, namely the tendency to learn lower-frequency components earlier or more easily than higher-frequency ones [47, 65].

For sequential models, inductive biases are especially central because sequence architectures differ sharply in how they encode time, locality, memory, and scale. A large body of work studies such biases either explicitly or implicitly [60, 14]. Among the most common examples are *recency bias*, where recent tokens or states receive disproportionate weight [27]; *frequency bias*, where models prefer smoother or lower-frequency temporal structure [59]; *locality bias*, where nearby tokens interact more strongly than distant ones [5]; *multi-scale or hierarchical bias*, where architectures favor representations that decompose across resolutions [10]; *position or order bias*, induced by causal masking or positional encodings [56]; and *state-tracking or finite-memory bias*, which governs how information is preserved across long horizons [54]. These biases appear in different forms across convolutions, recurrent models, Transformers, and SSMs, and they often interact rather than acting in isolation.

Compression of State-Space Models. The idea of compressing an SSM naturally connects to the classical literature on model order reduction (MOR), which seeks lower-order dynamical systems that preserve the input–output behavior of a higher-order model [4]. Canonical tools in this area include balanced truncation [23] and Hankel-norm approximation [16, 64], which come with strong approximation guarantees. A recurring tension, however, is that many of these methods rely on similarity transforms or dense reduced realizations that can destroy the diagonal or structured parameterizations exploited by modern SSMs for efficiency and stability. This challenge is already emphasized in recent SSM pruning work, which positions post-training compression as a form of MOR adapted to deep stacks and structured state spaces.

Recent work has therefore focused on structure-aware compression schemes for learned SSMs. LAST introduces layer-adaptive state pruning based on per-state $\mathcal{H}_\infty$ scores and a global cross-layer ranking, showing that trained SSMs often contain substantial removable redundancy [24]. AIRE-Prune replaces this worst-case perspective with an asymptotic impulse-response energy criterion, again in a post-training setting, and reports strong compression with little accuracy loss [42].

A complementary direction aims to compress SSMs during optimization rather than after training. For example, [7] uses Hankel-singular-value ideas to begin with a larger linear SSM and then reduce its dimension over the course of training, improving the tradeoff between expressivity and efficiency. Other orthogonal directions reduce cost by changing the inference mechanism rather than pruning states directly, for example, via transfer-function or state-free inference formulations [43]. Our approach is orthogonal to these lines of work: instead of reducing the state dimension, we compress the time axis during training, and can therefore be combined with state-pruning or in-training reduction methods.

B Proof of Theoretical Results

In this appendix, we present the proofs of our theoretical claims that S4 is a more continuous class of models than S6. To begin with, we prove Lemma 1 about a general continuous system, and the rest of the claims follow easily.

To prove Lemma 1, we leverage a key property of the ZOH discretization: if the function $u(t)$ is constant on each interval between the sampling points, then the ZOH discretization does not induce any error. In our assumption, however, $u(t)$ is obviously not piecewise constant, but we can approximate $u(t)$ by a piecewise-constant function $v(t)$. Now, by a triangle inequality, the difference between the continuous output on $u(t)$ and the discrete output on $u(\tau k)$ is upper-bounded by the sum of three terms:

- the difference between the continuous output on $u(t)$ and the continuous output on $v(t)$,
- that between the continuous output on $v(t)$ and the discrete output on $v(\tau k)$,
- and that between the discrete output on $u(\tau k)$ and the discrete output on $v(\tau k)$.

By defining v so that $u(\tau k) = v(\tau k)$, the last term always vanishes. Moreover, for the ZOH discretization, the second term is also zero. It therefore remains to control the first term, which reduces to controlling the growth of a system's states and output given a zero initial condition and a small input $u(t) - v(t)$. Hence, it naturally fits into a Grönwall-type analysis. We now formalize this argument.

Proof of Lemma 1. Fix a $\tau \in (0, 1)$, define a step function $v^\tau : [0, 1] \to \mathbb{R}$ by

$$v^\tau(t) = u(\tau \lfloor t/\tau \rfloor), \qquad t \in [0, 1].$$

Since $u(t)$ is L_u-Lipschitz continuous, we have

$$|u(t) - v^\tau(t)| \le L_u \tau, \qquad 0 \le t \le 1. \tag{7}$$

Define $\boldsymbol{\xi}^\tau(t)$ and $z^\tau(t)$ to be the solution of the continuous system on input v^τ, i.e.,

$$\begin{aligned} (\boldsymbol{\xi}^\tau)'(t) &= \Delta(v^\tau(t))(\mathbf{A}\boldsymbol{\xi}^\tau(t) + \mathbf{B}(v^\tau(t))v^\tau(t)), \\ z^\tau(t) &= \mathbf{C}(v^\tau(t))^\top \boldsymbol{\xi}^\tau(t) + Dv^\tau(t), \end{aligned} \qquad \boldsymbol{\xi}^\tau(0) = \mathbf{0}. \tag{8}$$

In what follows, we break the proof into two parts, corresponding to the case where ZOH or bilinear discretization is used, respectively.

CASE I: ZOH Discretization. Define $\mathbf{z}^\tau = (z_0^\tau, \dots, z_{\lfloor 1/\tau \rfloor}^\tau)$ to be the solution of the discrete system

$$\begin{aligned} \boldsymbol{\xi}_{k+1}^\tau &= \overline{\mathbf{A}}_k^\tau \boldsymbol{\xi}_k^\tau + \overline{\mathbf{B}}_k^\tau v(\tau k), \\ z_k^\tau &= (\overline{\mathbf{C}}_k^\tau)^\top \boldsymbol{\xi}_k^\tau + \overline{D} v(\tau k), \end{aligned} \qquad \boldsymbol{\xi}_0^\tau = \mathbf{0},$$

where $\overline{\mathbf{A}}_k^\tau$, $\overline{\mathbf{B}}_k^\tau$, $\overline{\mathbf{C}}_k^\tau$, and $\overline{D}$ are defined by the ZOH formula in eq. (2). Note that since we have $v(\tau k) = u(\tau k)$ for every $k = 0, \dots, \lfloor 1/\tau \rfloor$, we have

$$z_k^\tau = y_k^\tau, \qquad 1 \le k \le \lfloor 1/\tau \rfloor. \tag{9}$$

Moreover, the discrete system defined by the ZOH formula in eq. (2) also forms a ZOH discretization of the system in eq. (8). Since v^τ is a step function with a step size of τ, by definition, we have

$$z_k^\tau = z^\tau(\tau k), \qquad 1 \le k \le \lfloor 1/\tau \rfloor. \tag{10}$$

Next, define

$$\boldsymbol{\varepsilon}^\tau = \boldsymbol{\xi}^\tau - \mathbf{x} : [0, 1] \to \mathbb{R}.$$

We have

$$\begin{aligned} (\boldsymbol{\varepsilon}^\tau)'(t) &= \Delta(v^\tau(t))(\mathbf{A}\boldsymbol{\xi}^\tau(t) + \mathbf{B}(v^\tau(t))v^\tau(t)) - \Delta(u(t))(\mathbf{A}\mathbf{x}(t) + \mathbf{B}(u(t))u(t)) \\ &= \Delta(v^\tau(t))(\mathbf{A}\boldsymbol{\xi}^\tau(t) + \mathbf{B}(v^\tau(t))v^\tau(t)) \\ &\quad - (\Delta(v^\tau(t)) + (\Delta(u(t)) - \Delta(v^\tau(t))))(\mathbf{A}\mathbf{x}(t) + \mathbf{B}(u(t))u(t)) \\ &= \underbrace{\Delta(v^\tau(t))\mathbf{A}\boldsymbol{\varepsilon}^\tau(t)}_{\boldsymbol{\varepsilon}_1^\tau(t)} + \underbrace{\Delta(v^\tau(t))(\mathbf{B}(v^\tau(t))v^\tau(t) - \mathbf{B}(u(t))u(t))}_{\boldsymbol{\varepsilon}_2^\tau(t)} \\ &\quad + \underbrace{(\Delta(u(t)) - \Delta(v^\tau(t)))\mathbf{A}\mathbf{x}(t)}_{\boldsymbol{\varepsilon}_3^\tau(t)} + \underbrace{(\Delta(u(t)) - \Delta(v^\tau(t)))\mathbf{B}(u(t))u(t)}_{\boldsymbol{\varepsilon}_4^\tau(t)} \end{aligned}$$

We then control these four terms separately. First, we have

$$\|\boldsymbol{\varepsilon}_1^\tau(t)\|_2 \le |\Delta(v^\tau(t))|\|\mathbf{A}\|_2\|\boldsymbol{\varepsilon}^\tau(t)\|_2 \le M_\Delta\|\mathbf{A}\|_2\|\boldsymbol{\varepsilon}^\tau(t)\|_2, \qquad 0 \le t \le 1, \tag{11}$$

where we control $|\Delta(v^\tau(t))|$ using the fact that Range$(v) \subset$ Range(u). Next, we have

$$\begin{aligned}\|\boldsymbol{\varepsilon}_2^\tau(t)\|_2 &\le M_\Delta(\|\mathbf{B}(v^\tau(t)) - \mathbf{B}(u(t))\|_2|u(t)| + \|\mathbf{B}(v^\tau(t))\|_2|v^\tau(t) - u(t)|),\\ &\le M_\Delta(L_B L_u M_u \tau + M_B L_u \tau),\\ &= M_\Delta(L_B M_u + M_B)L_u\tau, \qquad 0 \le t \le 1.\end{aligned} \tag{12}$$

From

$$\mathbf{x}'(t) = \Delta(u(t))(\mathbf{A}\mathbf{x}(t) + \mathbf{B}(u(t))u(t)), \qquad \mathbf{x}(0) = 0,$$

we obtain

$$\|\mathbf{x}'(t)\|_2 \le M_\Delta\|\mathbf{A}\|_2\|\mathbf{x}(t)\|_2 + M_\Delta M_B M_u.$$

Hence, by Grönwall's inequality,

$$\|\mathbf{x}(t)\|_2 \le \frac{M_B M_u}{\|\mathbf{A}\|_2}\big(e^{M_\Delta\|\mathbf{A}\|_2} - 1\big), \qquad 0 \le t \le 1.$$

Therefore, the third term can be bounded by

$$\|\boldsymbol{\varepsilon}_3^\tau(t)\|_2 \le L_\Delta L_u\tau\|\mathbf{A}\|_2\|\mathbf{x}(t)\|_2 \le L_\Delta L_u\tau M_B M_u\left(e^{M_\Delta\|\mathbf{A}\|_2} - 1\right), \qquad 0 \le t \le 1. \tag{13}$$

The final term satisfies that

$$\|\boldsymbol{\varepsilon}_4^\tau(t)\|_2 \le L_\Delta L_u\tau M_B M_u, \qquad 0 \le t \le 1. \tag{14}$$

Now, combining eq. (11)-eq. (14), for every $0 \le t \le 1$, we have

$$\|(\boldsymbol{\varepsilon}^\tau)'(t)\|_2 \le M_\Delta\|\mathbf{A}\|_2\|\boldsymbol{\varepsilon}^\tau(t)\|_2 + \left[M_\Delta(L_B M_u + M_B)L_u + L_\Delta L_u M_B M_u e^{M_\Delta\|A\|_2}\right]\tau.$$

Since $\boldsymbol{\varepsilon}^\tau(0) = \mathbf{0}$, by Grönwall's inequality again, we have

$$\|\boldsymbol{\varepsilon}^\tau(t)\|_2 \le \frac{M_\Delta(L_B M_u + M_B)L_u + L_\Delta L_u M_B M_u e^{M_\Delta\|\mathbf{A}\|_2}}{M_\Delta\|\mathbf{A}\|_2}\,(\exp(M_\Delta\|\mathbf{A}\|_2 t) - 1)\tau.$$

Hence, for every $0 \le k \le \lfloor 1/\tau \rfloor$, by Hölder's inequality, we have

$$\begin{aligned}|z^\tau(\tau k) - y(\tau k)| &\le \|\mathbf{C}(u(k\tau))\|_2\|\boldsymbol{\varepsilon}^\tau(k\tau)\|_2\\ &\le M_C\frac{M_\Delta(L_B M_u + M_B)L_u + L_\Delta L_u M_B M_u e^{M_\Delta\|\mathbf{A}\|_2}}{M_\Delta\|\mathbf{A}\|_2}\,(\exp(M_\Delta\|\mathbf{A}\|_2) - 1)\tau.\end{aligned} \tag{15}$$

Combining eq. (9), eq. (10), and eq. (15), for every $0 \le k \le \lfloor 1/\tau \rfloor$, we have

$$\begin{aligned}|y_k^\tau - y(\tau k)| &\le |y_k^\tau - z_k^\tau| + |z_k^\tau - z^\tau(\tau k)| + |z^\tau(\tau k) - y(\tau k)|\\ &\le 0 + 0 + M_C\frac{M_\Delta(L_B M_u + M_B)L_u + L_\Delta L_u M_B M_u e^{M_\Delta\|\mathbf{A}\|_2}}{M_\Delta\|\mathbf{A}\|_2}\,(\exp(M_\Delta\|\mathbf{A}\|_2) - 1)\tau.\end{aligned}$$

This proves the result for the ZOH discretization.

CASE II: Bilinear Discretization. Define also the discrete trajectory $\mathbf{z}^\tau = (z_0^\tau, \ldots, z_{\lfloor 1/\tau \rfloor}^\tau)$ by the bilinear recursion

$$\begin{aligned}\boldsymbol{\xi}_{k+1}^\tau &= \overline{\mathbf{A}}_k^\tau\boldsymbol{\xi}_k^\tau + \overline{\mathbf{B}}_k^\tau v^\tau(\tau k),\\ z_k^\tau &= (\overline{\mathbf{C}}_k^\tau)^\top\boldsymbol{\xi}_k^\tau + \overline{D}\,v^\tau(\tau k),\end{aligned} \qquad \boldsymbol{\xi}_0^\tau = \mathbf{0},$$

where $\overline{\mathbf{A}}_k^\tau, \overline{\mathbf{B}}_k^\tau, \overline{\mathbf{C}}_k^\tau, \overline{D}$ are now defined by the bilinear formula in eq. (2). Since $v^\tau(\tau k) = u(\tau k)$ for every k, we again have

$$z_k^\tau = y_k^\tau, \qquad 0 \le k \le \lfloor 1/\tau \rfloor. \tag{16}$$

Unlike the ZOH case, the bilinear discretization is not exact on the step input v^τ, so in general $z_k^\tau \ne z(\tau k)$. We therefore decompose

$$|y_k^\tau - y(\tau k)| \le |y_k^\tau - z_k^\tau| + |z_k^\tau - z(\tau k)| + |z(\tau k) - y(\tau k)|. \tag{17}$$

The first term vanishes by eq. (16). The third term follows the same argument in the ZOH case and satisfies that

$$|z^\tau(\tau k) - y(\tau k)| \le M_C \frac{M_\Delta(L_B M_u + M_B)L_u + L_\Delta L_u M_B M_u e^{M_\Delta \|\mathbf{A}\|_2}}{M_\Delta \|\mathbf{A}\|_2} \left(e^{M_\Delta \|\mathbf{A}\|_2} - 1\right) \tau. \tag{18}$$

It remains to control $|z_k^\tau - z(\tau k)|$. Since v^τ is constant on each interval $[\tau k, \tau(k+1))$, the continuous dynamics on that interval reduce to the linear ODE

$$(\boldsymbol{\xi}^\tau)'(t) = \Delta(u(\tau k))\big(\mathbf{A}\boldsymbol{\xi}^\tau(t) + \mathbf{B}(u(\tau k))u(\tau k)\big),$$

that is, to a constant-coefficient affine system

$$(\boldsymbol{\xi}^\tau)'(t) = \mathbf{M}_k \boldsymbol{\xi}^\tau(t) + \mathbf{g}_k,$$

with

$$\mathbf{M}_k := \Delta(u(\tau k))\mathbf{A}, \qquad \mathbf{g}_k := \Delta(u(\tau k))\mathbf{B}(u(\tau k))u(\tau k).$$

The bilinear update is precisely the trapezoidal-rule discretization of this ODE. Since the exact solution of a constant-coefficient linear ODE is smooth, the trapezoidal rule has local truncation error of order $\mathcal{O}(\tau^3)$ on each step, uniformly in k. Moreover, the one-step maps are uniformly stable for sufficiently small τ, i.e.,

$$\left\| \left(\mathbf{I} - \frac{\tau}{2}\mathbf{M}_k\right)^{-1} \left(\mathbf{I} + \frac{\tau}{2}\mathbf{M}_k\right) \right\|_2 \le 1 + C\tau, \qquad 0 \le k \le \lfloor 1/\tau \rfloor,$$

for some constant $C > 0$ independent of k. Therefore, by the standard global error estimate for the trapezoidal rule, there exists a constant $C' > 0$, independent of τ and k, such that

$$\|\boldsymbol{\xi}_k^\tau - \boldsymbol{\xi}^\tau(\tau k)\|_2 \le C'\tau^2, \qquad 0 \le k \le \lfloor 1/\tau \rfloor. \tag{19}$$

Consequently,

$$|z_k^\tau - z(\tau k)| = |(\mathbf{C}(u(\tau k)))^\top (\boldsymbol{\xi}_k^\tau - \boldsymbol{\xi}^\tau(\tau k))| \le M_C \|\boldsymbol{\xi}_k^\tau - \boldsymbol{\xi}^\tau(\tau k)\|_2 \le M_C C'\tau^2. \tag{20}$$

Combining eqs. (17), (18) and (20), for every $0 \le k \le \lfloor 1/\tau \rfloor$, we obtain

$$|y_k^\tau - y(\tau k)| \le M_C \frac{M_\Delta(L_B M_u + M_B)L_u + L_\Delta L_u M_B M_u e^{M_\Delta \|\mathbf{A}\|_2}}{M_\Delta \|\mathbf{A}\|_2} \left(e^{M_\Delta \|\mathbf{A}\|_2} - 1\right) \tau + \mathcal{O}(\tau^2).$$

This proves the desired estimate. □

Now that we have proved Lemma 1, showing the corollaries on S4 and S6 is a simple task to plug in the maximum moduli and the Lipschitz constants into the lemma. This is particularly simple for S4 because the parameters are input-independent, making the Lipschitz constants $L_B = L_C = L_\Delta = 0$. For S6, the Lipschitz constants can be large, making the discretization errors usually larger.

Proof of Corollary 1. Since $\mathbf{B}, \mathbf{C}$, and Δ are constant, we have $L_B = L_C = L_\Delta = 0$. Thus, plugging in the expressions gives us

$$\begin{aligned} &M_C \frac{M_\Delta(L_B M_u + M_B)L_u + L_\Delta L_u M_B M_u e^{M_\Delta \|\mathbf{A}\|_2}}{M_\Delta \|\mathbf{A}\|_2} \left(\exp(M_\Delta \|\mathbf{A}\|_2) - 1\right) \\ &\quad = \frac{\|\mathbf{C}\|_2 \|\mathbf{B}\|_2 \Delta L_u (\exp(\Delta \|\mathbf{A}\|_2) - 1)}{\Delta \|\mathbf{A}\|_2} = \frac{\|\mathbf{C}\|_2 \|\mathbf{B}\|_2 L_u (\exp(\Delta \|\mathbf{A}\|_2) - 1)}{\|\mathbf{A}\|_2}. \end{aligned}$$

This proves the corollary. □

Proof of Corollary 2. By definition, we have

$$L_B = \|\mathbf{W_B}\|_2, \qquad L_C = \|\mathbf{W_C}\|_2, \qquad L_\Delta = |w_\Delta|.$$

Moreover, computing the maximum moduli of $\mathbf{B}, \mathbf{C}$, and Δ gives us

$$M_B = \|\mathbf{W_B}\|_2 M_u, \qquad M_C = \|\mathbf{W_C}\|_2 M_u, \qquad M_\Delta = \text{softplus}(|w_\Delta| M_u + b_\Delta).$$

Plugging these expressions into Lemma 1, we obtain

$$
\begin{aligned}
&M_C \frac{M_\Delta(L_B M_u + M_B)L_u + L_\Delta L_u M_B M_u e^{M_\Delta \|\mathbf{A}\|_2}}{M_\Delta \|\mathbf{A}\|_2} \left(e^{M_\Delta \|\mathbf{A}\|_2} - 1\right) \\
&\leq \|\mathbf{W_C}\|_2 M_u \frac{M_\Delta(\|\mathbf{W_B}\|_2 M_u + \|\mathbf{W_B}\|_2 M_u)L_u + |w_\Delta| L_u \|\mathbf{W_B}\|_2 M_u^2 e^{M_\Delta \|\mathbf{A}\|_2}}{M_\Delta \|\mathbf{A}\|_2} \left(e^{M_\Delta \|\mathbf{A}\|_2} - 1\right) \\
&= \frac{\|\mathbf{W_C}\|_2 \|\mathbf{W_B}\|_2 M_u^2 L_u \left(2M_\Delta + |w_\Delta| M_u e^{M_\Delta \|\mathbf{A}\|_2}\right)}{M_\Delta \|\mathbf{A}\|_2} \left(e^{M_\Delta \|\mathbf{A}\|_2} - 1\right).
\end{aligned}
$$

Finally, substituting

$$M_\Delta = \text{softplus}(|w_\Delta| M_u + b_\Delta)$$

yields the claimed estimate. □

C Details of Experiments in Section 3

In this appendix, we present the details of the experiments in the main text that aim to understand the convergence of S4 and S6's discrete outputs to their underlying continuous outputs, respectively. There are two experiments: a numerical one and a synthetic training experiment on dynamical systems. We explain them separately.

C.1 The numerical experiment

In the experiment demonstrated in Figure 3, we consider single-input, single-output dynamical systems. We randomly sample 20 system–input pairs as follows. First, we generate the input by forming a random linear combination of the first 20 Chebyshev polynomials on $[0, 1]$:

$$u(t) = \sum_{j=1}^{20} C_j T_j(t), \qquad C_j \sim \mathcal{N}(0, 1), \qquad 1 \le j \le 20,$$

where T_j is the j-th order Chebyshev polynomial basis function. To generate a random system, we let $\mathbf{A}$ be the diagonal HiPPO-LegS matrix used in many SSMs (see [19]) and randomly sample

$$\mathbf{B} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_n), \qquad \mathbf{C} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_n),$$

where $n = 8$ is the number of hidden states. Since the D term does not induce a discretization error, we set it to zero. The matrices $\mathbf{B}$ and $\mathbf{C}$ are then used to form an S4 and an S6 system: for S4, they are directly used as the parameters of the LTI system; for S6, the matrices $\mathbf{B}$ and $\mathbf{C}$ are the parameters that define the projection maps $\mathbf{B}(\cdot)$ and $\mathbf{C}(\cdot)$. For S6, we additionally define the weights and biases in the gating term $\Delta(u)$:

$$w_\Delta \sim \mathcal{N}(0, 1), \qquad b_\Delta = \text{softplus}^{-1}(0.01),$$

and for S4, we define the constant sampling interval to be $\Delta = 0.01$, which is consistent with b_Δ used in S6. To obtain an accurate benchmark for the continuous-level output, we set $\tau_0 = 2^{-14}$ and use the fourth-order Runge–Kutta method to simulate the S4 and S6 systems. We then simulate the two systems using τ ranging from 2^{-10} to 2^{-2} and compute the L_∞ errors. For a new scaling factor, we multiply it to the input function $u(t)$ and repeat the experiment. The experiments are repeated for 20 different input $u(t)$ and S4/S6 systems. All numerical simulations are carried out using MATLAB R2025b.

C.2 Experiment on dynamical systems

In the next experiment that we present in Figure 4, we consider four standard parameterized dynamical systems. In each case, the model receives a one-dimensional observed time series and predicts the underlying system parameters. We first explain these four classes of dynamical systems and then discuss the experimental setup.

Van der Pol oscillator. The Van der Pol oscillator is a classical nonlinear system originally introduced to model self-sustained oscillations in vacuum-tube circuits. For $\mu > 0$, trajectories converge to a stable limit cycle; small μ yields near-sinusoidal oscillations, while large μ produces relaxation oscillations with pronounced slow–fast structure. The dynamics are

$$\ddot{x} - \mu(1 - x^2)\dot{x} + x = 0,$$

or equivalently, in first-order form with state (x, v),

$$\dot{x} = v, \qquad \dot{v} = \mu(1 - x^2)v - x.$$

In our setting, the model observes $x(t)$ and predicts the scalar nonlinearity parameter μ.

Damped harmonic oscillator. The damped harmonic oscillator is a standard linear model for oscillatory motion with energy loss, appearing in mechanical vibrations and electrical RLC circuits. Its behavior is governed by the natural frequency ω and damping ratio ζ, which determine oscillation frequency and decay rate. The dynamics are

$$\ddot{x} + 2\zeta\omega\,\dot{x} + \omega^2 x = 0,$$

or in first-order form,

$$\dot{x} = v, \qquad \dot{v} = -2\zeta\omega\, v - \omega^2 x.$$

In our setting, the model observes $x(t)$ and predicts the parameters (ω, ζ).

Ornstein–Uhlenbeck (OU) process. The OU process is a canonical continuous-time stochastic process with mean-reverting behavior, widely used in statistical physics, e.g., Langevin dynamics, and finance, e.g., interest-rate models. Unlike deterministic systems, OU trajectories exhibit intrinsic randomness due to process noise. The dynamics are given by the SDE

$$dX_t = -\theta X_t \, dt + \sigma \, dW_t,$$

where W_t is standard Brownian motion, $\theta > 0$ controls the mean-reversion rate, and $\sigma \geq 0$ controls the diffusion strength. In our setting, the model observes X_t and predicts (θ, σ).

Forced Duffing oscillator. The forced Duffing oscillator is a prototypical nonlinear driven system exhibiting multistability and, for suitable parameters, chaos. It is commonly used as a benchmark for nonlinear system identification because small parameter changes can substantially alter long-term behavior, especially in chaotic regimes. The dynamics are

$$\ddot{x} + \delta \dot{x} + \alpha x + \beta x^3 = \gamma \cos(\omega t),$$

or equivalently,

$$\dot{x} = v, \qquad \dot{v} = -\delta v - \alpha x - \beta x^3 + \gamma \cos(\omega t).$$

In our setting, the model observes $x(t)$ and predicts the scalar forcing amplitude γ, while holding $(\delta, \alpha, \beta, \omega)$ fixed.

For each class of dynamical systems, we generate 10,000 random training trajectories using a double-precision Runge–Kutta integrator with a sampling interval of $\tau = 0.01$. Note that even if for some training tasks, one only has one or two undetermined parameters, we also randomly sample the zeroth-order and first-order initial conditions, making the training trajectories heterogeneous enough. We then generate 1,000 random validation trajectories. For each training task, we use a one-layer model with a hidden dimension of $d = 64$ and a latent-state dimension of $n = 32$. All models are trained for 10 epochs. For every model, we sweep the same ten learning rates between 10^{-3} and 10^{-2} and keep the checkpoint with the lowest validation loss; no other hyperparameters are tuned, so all model classes receive the same tuning effort and training budget.

D Details of Experiments in Section 4

In this appendix, we present the details of the experiments in the main text that aim to understand the relationship between the continuity of a model and the continuity of a dataset.

For the synthetic dynamical systems dataset, we combine the four classes of dynamical systems discussed in section 3. The goal, however, is to predict the next sample in the trajectory, as opposed to unraveling the coefficients in the dynamical systems. The model returns its prediction of the embedded token, instead of the raw data. That is, given a sampled sequence

$$u(0), u(\tau), \ldots, u(\tau L)$$

and its embedding

$$\mathbf{U}_\eta(0), \mathbf{U}_\eta(\tau), \ldots, \mathbf{U}_\eta(\tau L),$$

the model outputs its prediction of $\mathbf{U}_\eta(\tau(L+1))$ instead of $u(\tau(L+1))$. We made this decision because quantization inevitably takes away some precision from the input data, making a high-precision prediction of $u(\tau(L+1))$ intrinsically harder than that of $\mathbf{U}_\eta(\tau(L+1))$.

We train an S4, an S6, and a Transformer for 10 epochs. For every model, we set the hidden dimension to be $d = 128$ and the number of layers to be 4. In addition, the Transformer has 8 heads and the two SSMs have a state-space dimension of $n = 32$. As in Appendix C, every model is tuned over the same ten learning rates and selected by validation loss.

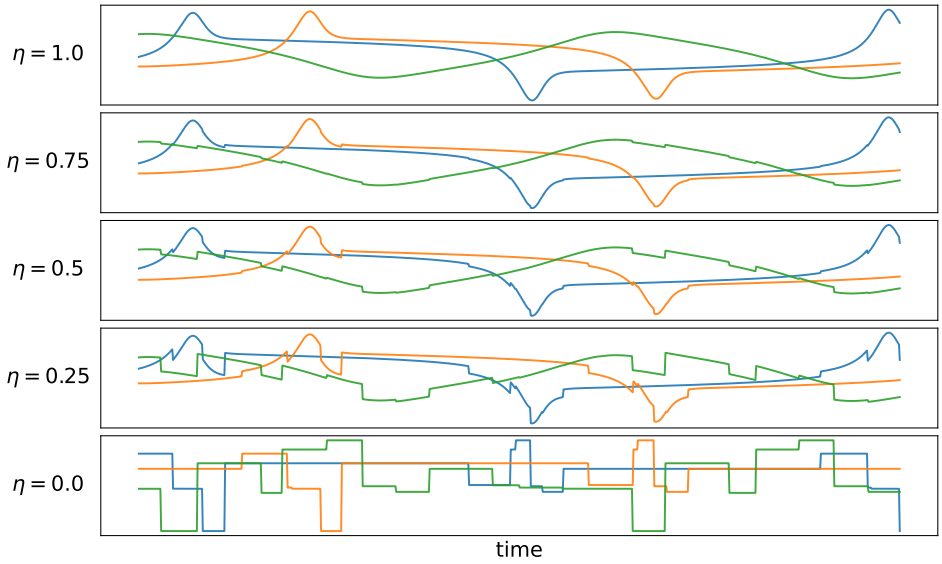


Figure 7: Visualization of three trajectories from a Van der Pol oscillator. Given a different η, the input is obtained by a weighted combination of a continuously embedded trajectory and a discretely embedded trajectory. As η goes from 1 to 0, the level of the continuity of the task visually decreases.

In Figure 7, we present a visualization of three embedded trajectories $\mathbf{U}_\eta(t)$ from a Van der Pol oscillator, as η ranges from 1 to 0. The visualization is done on the first channel of the 16-dimensional input data $\mathbf{U}_\eta(t)$. Clearly, one can also visualize the data by performing a principal component analysis (PCA); however, since each $\mathbf{U}_\eta(t)$ is a sampled column of a fixed orthogonal matrix when $\eta = 0$ and $\mathbf{U}_\eta(\cdot)$ is a rank-one quasimatrix when $\eta = 1$, a PCA would help in neither case.

Looking at the two metrics η and μ that we adopt in section 4, you may also wonder: are they consistent with each other? Clearly, given any general dataset, one can compute a μ but not an η, but for the dynamical systems dataset, one can still compute the μ-metric to see if μ indeed positively correlates to η. In Figure 8, we present this analysis. We can see that for each lag parameter t, η and μ_t are positively correlated. This further shows that our proposed μ-metric is consistent with our intuition in the continuity of a dataset.

Finally, we explain how the data in Figure 5 are obtained. For S4D, we directly collect data from the original S4D paper [19]; for Transformers, experiments show that training a Transformer from

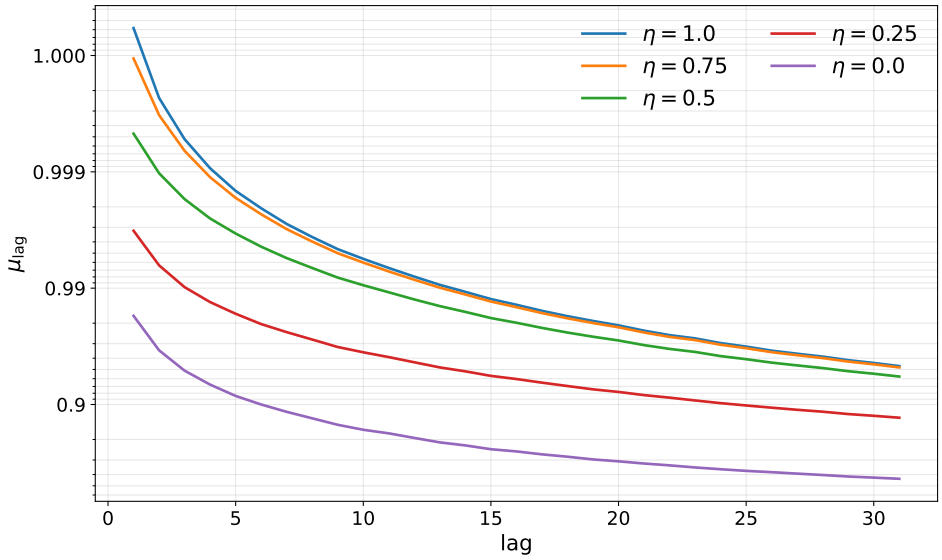


Figure 8: The metric μ_{lag} that we proposed in section 4 computed on the dynamical systems data embedded with a different level of continuity η. As η decreases from 1 to 0, the metric μ_{lag} also decreases. This shows a consistency between the intuitive continuity captured by η and the numerical continuity measured by our proposed μ_{lag} metric.

scratch on LRA tasks does not lead to good performance, and we therefore adopt data induced by the pretraining method proposed in [2]. For S6, [58] shows that the performance on LRA can be significantly strengthened by initializing the weight w_Δ small, adopting a multihead-plus-bias structure, and using a complex parameterization. Many of these approaches can be viewed as a way to make the model more continuous; therefore, we do not adopt any of these methods. Instead, the data for S6 are obtained by maximizing the per-task performance reported in [1], [49], [37], and of the vanilla Mamba pretrained using the method proposed in [2].

Table 2: Normalized relative gaps in the dynamical system forecasting error across continuity levels η, with cell colors computed from row-normalized values so that each model has comparable total mass across the row while preserving within-row proportions. We see that the better predictions happen at the lower-left part of the table, where both the model and the problem are discrete, and at the upper-right part of the table, where both the model and the problem are continuous.

	0	0.1	0.25	0.4	0.5	0.6	0.75	0.9	1
S4	0.630	0.560	0.418	0.112	0.000	0.000	0.000	0.000	0.000
DeltaNet	0.494	0.450	0.271	0.114	0.033	0.093	0.098	0.083	0.083
Transformer	0.299	0.368	0.319	0.114	0.064	0.116	0.140	0.131	0.169
LinAttn	0.029	0.225	0.256	0.161	0.261	0.283	0.127	0.169	0.209
S6	0.137	0.196	0.335	0.150	0.105	0.161	0.203	0.181	0.253
Mamba3	0.000	0.000	0.000	0.000	0.281	0.254	0.387	0.245	0.553

In Table 2, we repeat the experiment in Figure 5 with three additional models: Linear Attention [29], DeltaNet [57], and Mamba-3 [31]. We report the "normalized relative gap." Specifically, given a model $\mathcal{M}$ and a continuity level η, let $e^\eta_{\mathcal{M}}$ denote the $\mathcal{L}_\infty$ forecasting error of $\mathcal{M}$ averaged over the test set. Let $\mathbb{M}$ denote the collection of all six models under consideration. We first define the relative gap of model $\mathcal{M}$ at continuity level η by

$$\text{gap}^\eta_{\mathcal{M}} = \frac{e^\eta_{\mathcal{M}} - \min_{\mathcal{M}' \in \mathbb{M}} e^\eta_{\mathcal{M}'}}{\min_{\mathcal{M}' \in \mathbb{M}} e^\eta_{\mathcal{M}'}}.$$

This quantity measures how far $\mathcal{M}$ is from the best-performing model at the same continuity level η. However, this metric still has one limitation when used to compare different architectures: each model comes with its own parameter count, architecture class, and inductive biases beyond continuity, so some models may be uniformly stronger or weaker across all values of η. As a result, the raw relative gap is appropriate for tracking how a fixed architecture responds to changing continuity, but it is less suitable for comparing the shape of this trend across architectures. To remove this architecture-dependent scale effect while preserving the relative variation of each model across continuity levels, we introduce the normalized gap

$$\overline{\text{gap}}^{\eta}_{\mathcal{M}} = \alpha_{\mathcal{M}} \, \text{gap}^{\eta}_{\mathcal{M}}, \qquad \alpha_{\mathcal{M}} = \frac{\frac{1}{|\mathbb{M}|} \sum_{\widetilde{\mathcal{M}} \in \mathbb{M}} \sum_{\eta'} \text{gap}^{\eta'}_{\widetilde{\mathcal{M}}}}{\sum_{\eta'} \text{gap}^{\eta'}_{\mathcal{M}}}.$$

Equivalently, $\overline{\text{gap}}^{\eta}_{\mathcal{M}}$ is obtained by multiplying each row of relative gaps by a model-dependent constant so that all models have the same total gap mass across continuity levels. This normalization preserves the ratios between any two entries within the same row, and hence preserves how performance varies with continuity for each model, while making the rows more comparable across architectures.

E Details of Experiments in Section 5

In this appendix, we present the details of the experiments in the main text that aim to show how continuity can be leveraged to enhance the training efficiency of an S4D model on the LRA benchmark.

In Figure 6, we showed the time–accuracy plot for a vanilla S4D and a subsampled S4D trained on all six LRA benchmark tasks. There, we report the test accuracy of the best model, measured by

$$\text{test acc of the best model at epoch } K = \text{test acc at epoch } \{\arg\max_{i \le K}(\text{validation acc at epoch } i)\}.$$

In Figure 9, we further show the per-epoch test accuracy of the model, which gives us a more concrete understanding of how the stages transition. From Figure 9, we note an important finding:

> For the more continuous tasks, such as Image, Pathfinder, and PathX, the loss of test accuracy is relatively low at each stage transition; for the more discrete tasks, such as ListOps, Retrieval, and Text, the loss of accuracy is sharp at each stage transition. This shows that for the continuous tasks, more information is inherited by the refined model at the start of each training stage.

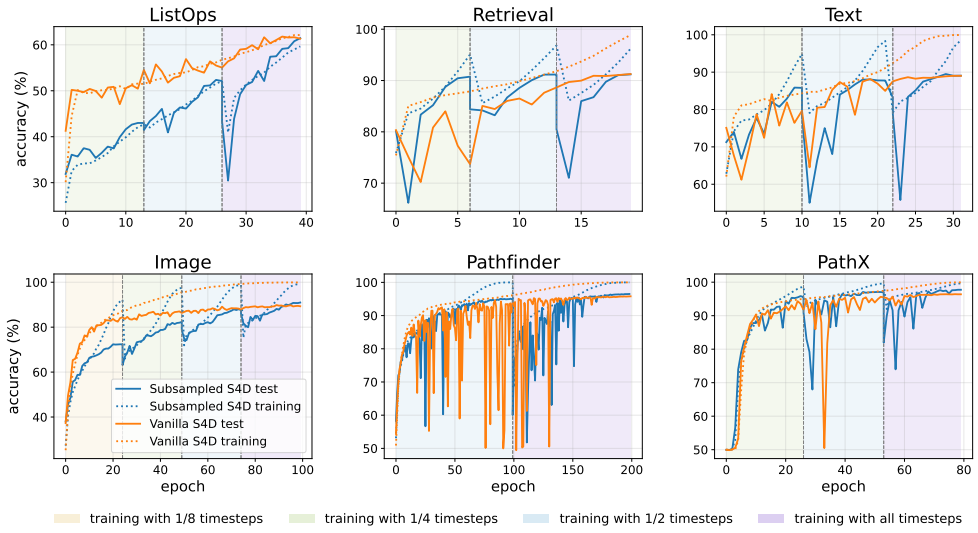


Figure 9: The per-epoch training and test accuracies of a naively trained S4D model and an S4D model trained with our subsampling method, respectively. We use a colorful background to indicate stages where models are trained on data with different temporal densities. Note that the background colors only apply to the models trained with the subsampling method, and that the sparser the timestamps are, the faster the training of the subsampled model is.

More numerical results can be found in Table 3. For training the subsampled models, there are a few hyperparameters that we need to choose. We briefly explain them:

- `Initial Density`: This indicates the temporal density of the data we subsample at the beginning of the training. For example, if the full data has a length of $L = 1024$, and the training data we use in epoch zero has a length of 128, then the initial density is $1/8$.
- `Subsampling Strategy`: This is a binary decision to make from "indexing" or "pooling." Assuming that the initial density is $1/r$ If the subsampling is done by indexing, then given a sequence $\mathbf{u}_0, \mathbf{u}_1 \dots, \mathbf{u}_{L-1}$, the subsampled sequence is $\mathbf{u}_0, \mathbf{u}_r, \dots, \mathbf{u}_{\lfloor (L-1)/r \rfloor r}$. If the subsampling is done by pooling, then the subsampled sequence is $\mathbf{v}_0, \mathbf{v}_r, \dots, \mathbf{v}_{\lfloor (L-1)/r \rfloor r}$, where

$$\mathbf{v}_j = \frac{1}{r} \sum_{i=0}^{r-1} \mathbf{u}_{jr+i}.$$

Given an initial density, which we always choose to be an inverse power of 2, we then evenly divide all training epochs into the resulting number of stages. While we only examine this way of division,

Table 3: Relative time needed to reach target test accuracies on Long-Range Arena. Each numeric entry reports the ratio between the time needed by the subsampled S4D model and the time needed by the vanilla S4D model. Values below 1 indicate faster convergence for the subsampled model.

(a) ListOps

Acc.	42	44	46	48	50
Ratio	5.60	6.49	7.15	7.23	6.67
Acc.	52	54	56	58	60
Ratio	1.07	1.61	1.13	0.83	0.80

(b) Retrieval

Acc.	81	82	83	84	85
Ratio	0.15	0.16	0.17	0.19	0.14
Acc.	86	87	88	89	90
Ratio	0.12	0.10	0.10	0.10	0.10

(c) Text

Acc.	77	78	79	80	81
Ratio	0.49	0.59	0.48	0.49	0.49
Acc.	82	83	84	85	86
Ratio	0.49	0.62	0.62	0.30	0.59

(d) Image

Acc.	40	45	50	55	60
Ratio	0.18	0.21	0.26	0.23	0.36
Acc.	65	70	75	80	85
Ratio	0.45	0.47	0.76	0.71	0.57

(e) Pathfinder

Acc.	59	63	67	71	75
Ratio	0.35	0.40	0.40	0.43	0.49
Acc.	79	83	87	91	95
Ratio	0.52	0.76	0.73	0.69	0.43

(f) PathX

Acc.	51	56	61	66	71
Ratio	0.16	0.16	0.17	0.17	0.18
Acc.	76	81	86	91	96
Ratio	0.19	0.20	0.22	0.23	0.19



subsampled time / vanilla time

other division strategies can be potentially interesting to explore. The pooling strategy is especially useful when the underlying task is more discrete than continuous, for otherwise a subsampling by index may totally destroy the useful information in the data. In fact, among the six tasks in LRA, only the most continuous one (i.e., Image; see Figure 5) can be trained using the indexing subsampling strategy. We report the hyperparameters that we use in Table 4. The subsampled runs inherit exactly the same model and optimization hyperparameters as the vanilla runs, with no additional tuning.

Table 4: The hyperparameters `Initial Density` and `Subsampling Strategy` used in training the S4D models on Long-Range Arena tasks. The rest of the hyperparameters follow those reported in the original S4D paper [19].

Hyperparameter	ListOps	Retrieval	Text	Image	Pathfinder	PathX
`Initial Density`	1/4	1/4	1/4	1/8	1/2	1/4
`Subsampling Strategy`	pooling	pooling	pooling	indexing	pooling	pooling

Finally, to show that the subsampling method is helpful only when the underlying model is continuous, we show in Figure 10 the results of applying the subsampling method on an S6 model trained on the Image task from LRA, in which case we see that most learned information is lost at each stage transition and the subsampling method does not help.

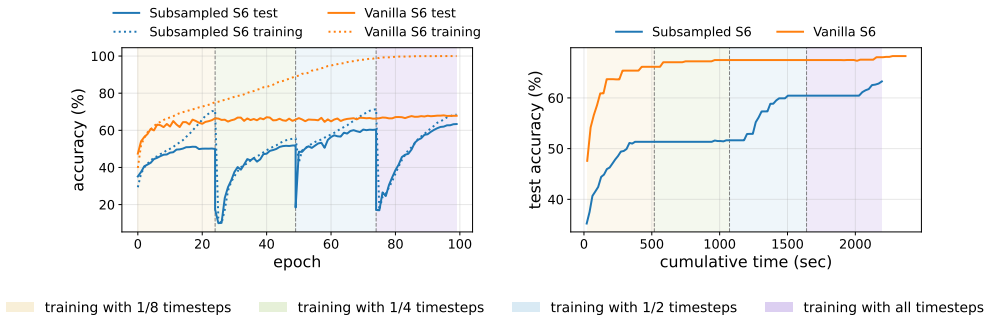


Figure 10: **Left:** The per-epoch training and test accuracies on the sCIFAR-10 task of a naively trained S6 model and an S6 model trained with our subsampling method, respectively. **Right:** Test accuracy plotted against cumulative training time on the sCIFAR-10 task. For more details, see the captions of Figures 6 and 9.

F Additional Experiments

This appendix collects the additional experiments that we ran during the review process. They sharpen a few points from the main text: how the relative error of S6 behaves when the input is scaled, how robust the task-continuity score is to its ingredients, how the ordering between S6 and Transformers depends on their hyperparameters, what happens when we make S6 more continuous by hand, how stable the stage-wise training results are across random seeds, and how S4 and S6 cope with missing observations on real-world time series.

F.1 Relative error under input scaling

Corollaries 1 and 2 bound the absolute error $|y_k^\tau - y(\tau k)|$. Since the constant for S6 shrinks with the input amplitude M_u, one may hope that S6 becomes more continuous on small inputs. The quantity that matters, however, is the relative error $|y_k^\tau - y(\tau k)|/|y(\tau k)|$, which is also what Figure 3 reports. Fix an input u and scale it by a factor $c > 0$. For S4, the system is linear, so both the output and the bound in Corollary 1 scale like c, and the relative error is $\mathcal{O}(1)$ as $c \to 0^+$ and as $c \to \infty$. For S6 (with $D = 0$, as in our numerical experiment), the input enters cubically: twice through $\mathbf{B}(u)u = \mathbf{W}_\mathbf{B}u^2$ and once more through $\mathbf{C}(u) = \mathbf{W}_\mathbf{C}u$. Hence, as $c \to 0^+$, the output scales like c^3, and so does the bound in Corollary 2, because $M_\Delta \to \text{softplus}(b_\Delta)$ stays bounded; the relative error is again $\mathcal{O}(1)$. As $c \to \infty$, however, $M_\Delta = \text{softplus}(|w_\Delta|M_u + b_\Delta)$ grows with c and the exponential factors in Corollary 2 take over, so the relative error grows. In short, shrinking the input does not help S6, while enlarging it hurts. Table 5 confirms this numerically in the setting of Figure 3.

Table 5: Relative discretization error of S4 and S6 at the fixed step size $\tau = 2^{-9}$ in the setting of Figure 3, as the input is scaled by factors from $1/32$ to 32. Shrinking the input does not reduce the relative error of S6, while enlarging it does increase the error.

Scale	$1/32$	$1/16$	$1/8$	$1/4$	$1/2$	1	2	4	8	16	32
S4	0.0025	0.0025	0.0025	0.0025	0.0025	0.0025	0.0025	0.0025	0.0025	0.0025	0.0025
S6	0.0024	0.0023	0.0024	0.0027	0.0026	0.0026	0.0049	0.0067	0.0084	0.0091	0.0103

F.2 Robustness of the task-continuity score

The score μ_t depends on a kernel, an encoder, and a lag range. To check that our conclusions do not hinge on these choices, we recompute μ_t on the six LRA tasks for lags $t \in \{1, 2, 4, 8, 16, 32, 64\}$ with four kernels: the cosine kernel used in the main text, which measures alignment; the RBF and Laplacian kernels, which measure Euclidean distance; and the Cauchy kernel, which is also distance-based but has a heavier tail. Table 6 reports the raw scores; unlike in Figure 5, they are not shifted to be positive for a logarithmic axis. A small negative value simply means that elements t positions apart are slightly less similar than the far-apart baseline, which is unsurprising for text and symbolic sequences, where neighboring tokens tend to differ. The qualitative ordering is the same for all four kernels: Image is the most continuous task, Pathfinder and PathX are intermediate, and ListOps, Retrieval, and Text sit near zero. (For Image and Pathfinder, lags 32 and 64 give high scores because they land on the same column of a 32×32 image, one or two rows below.) Table 7 repeats the computation with the encoders of pretrained S6 and Transformer models, as well as the more continuous S6 variants of Appendix F.4. The ordering of the tasks is again unchanged, and making S6 more continuous does not make the tasks look more continuous.

F.3 Sharper attention and stronger selectivity

In Figure 5 (Left), Transformers land between S4 and S6, although one might expect them to be the most discrete of the three. Their behavior depends on how sharply attention selects tokens. We therefore scale the attention logits by a temperature T, so that smaller T makes attention sharper and larger T spreads weight across more tokens, with $T = 1$ recovering the standard model. Likewise, for S6 we use $\Delta(u) = \text{softplus}(\alpha w_\Delta u + b_\Delta)$, so that smaller α weakens the input dependence of Δ, with $\alpha = 1$ recovering standard S6. Table 8 reports the forecasting error gap of every variant to the best of the three standard models. The ordering between S6 and the Transformer changes with α

Table 6: Task-continuity score μ_t on the six LRA tasks for four kernels and seven lags, together with the mean over these lags, computed with the encoders of pretrained S4 models. The scores are raw, i.e., not shifted to be positive as in Figure 5; small negative values mean that elements t positions apart are slightly less similar than far-apart elements. The ordering of the tasks is the same for every kernel.

Task	Lag 1	Lag 2	Lag 4	Lag 8	Lag 16	Lag 32	Lag 64	Mean
Cosine kernel								
ListOps	0.0022	0.0000	−0.0075	−0.0052	−0.0043	−0.0014	−0.0016	−0.0026
Retrieval	−0.0552	−0.0397	0.0066	0.0029	0.0044	0.0038	0.0028	−0.0106
Text	−0.0422	−0.0324	0.0019	0.0027	0.0025	0.0035	0.0027	−0.0088
Image	0.8819	0.7436	0.5721	0.3888	0.2670	0.8737	0.7121	0.6342
Pathfinder	0.4427	0.1217	0.0451	−0.0010	−0.0499	0.3959	0.0470	0.1431
PathX	0.6273	0.2101	−0.0028	−0.0330	−0.0107	−0.0221	−0.0148	0.1077
RBF kernel								
ListOps	0.0030	0.0006	−0.0029	−0.0056	−0.0017	−0.0002	−0.0023	−0.0013
Retrieval	−0.0543	−0.0410	0.0021	−0.0003	−0.0018	0.0000	0.0004	−0.0136
Text	−0.0440	−0.0328	0.0009	0.0020	0.0021	0.0007	0.0025	−0.0098
Image	0.7784	0.6157	0.4523	0.2935	0.2012	0.7604	0.5742	0.5251
Pathfinder	0.1431	0.1068	0.0413	0.0023	−0.0274	0.1212	0.0723	0.0657
PathX	0.3987	0.2154	0.0361	−0.0066	−0.0312	−0.0045	−0.0200	0.0840
Laplacian kernel								
ListOps	0.0037	−0.0002	−0.0015	−0.0039	−0.0022	0.0003	0.0007	−0.0004
Retrieval	−0.0535	−0.0387	0.0046	−0.0006	0.0015	0.0014	0.0026	−0.0118
Text	−0.0429	−0.0312	0.0018	0.0039	0.0031	0.0035	0.0036	−0.0083
Image	0.6478	0.5010	0.3611	0.2285	0.1536	0.6219	0.4563	0.4243
Pathfinder	0.1475	0.0979	0.0412	0.0020	−0.0251	0.1255	0.0659	0.0650
PathX	0.3664	0.2113	0.0136	0.0055	−0.0162	−0.0307	0.0039	0.0791
Cauchy kernel								
ListOps	0.0028	−0.0021	−0.0036	−0.0057	−0.0001	−0.0015	−0.0009	−0.0016
Retrieval	−0.0555	−0.0409	0.0034	−0.0008	−0.0015	−0.0002	0.0035	−0.0132
Text	−0.0423	−0.0330	0.0023	0.0026	0.0023	0.0019	0.0031	−0.0090
Image	0.7747	0.6162	0.4522	0.2887	0.1884	0.7570	0.5688	0.5208
Pathfinder	0.1491	0.0959	0.0350	0.0021	−0.0309	0.1302	0.0613	0.0632
PathX	0.4066	0.2316	0.0231	0.0019	−0.0070	0.0057	−0.0010	0.0944

Table 7: Task-continuity score (cosine kernel, mean over the seven lags of Table 6) computed with the encoders of different pretrained models, including the S6 variants of Appendix F.4 with weakened ($\alpha = 0.1$) or no ($\alpha = 0$) input dependence in Δ. The ordering of the tasks is stable, and making S6 more continuous does not change how continuous the tasks look.

	Encoder of a pretrained model				
Task	S4	S6 ($\alpha = 1$)	Transformer	S6 ($\alpha = 0.1$)	S6 ($\alpha = 0$)
ListOps	−0.0026	−0.0019	−0.0020	−0.0027	−0.0018
Retrieval	−0.0106	−0.0090	−0.0127	−0.0172	−0.0080
Text	−0.0088	−0.0122	−0.0075	−0.0048	−0.0131
Image	0.6342	0.6782	0.6499	0.6633	0.6419
Pathfinder	0.1431	0.1490	0.1378	0.1678	0.1394
PathX	0.1077	0.1214	0.0986	0.0899	0.1025

and T, so it is not universal; the configurations used in the main text are the standard ones, not the winners of a sweep. The main trend, however, is stable: S4 is favored as η increases, while the S6 and Transformer variants are favored in the more discrete regimes, and even the extreme choices of α and T do not remove this crossover.

Table 8: Relative L_∞ forecasting error gap (in %) to the best of the three standard models of Figure 5 (S4, S6 with $\alpha = 1$, and the Transformer with $T = 1$) at each continuity level η, for S6 with different strengths α of input dependence and Transformers with different attention temperatures T. Negative entries mean that a variant beats all three standard models. The crossover in favor of S4 as η grows persists for every choice of α and T.

η	0	0.1	0.25	0.4	0.5	0.6	0.75	0.9	1
S4	47.57	32.66	5.00	0.00	0.00	0.00	0.00	0.00	0.00
S6 ($\alpha = 0$)	46.92	44.55	11.28	8.72	−4.50	12.30	17.91	5.68	5.08
S6 ($\alpha = 0.5$)	13.18	15.18	11.29	16.06	12.37	17.48	20.67	22.17	19.37
S6 ($\alpha = 1$, standard)	0.00	0.00	0.00	4.78	12.86	19.76	24.83	22.12	30.97
S6 ($\alpha = 2$)	−1.80	−5.66	4.70	21.52	9.90	15.59	28.81	24.41	26.97
S6 ($\alpha = 4$)	−0.70	1.77	−2.52	17.62	11.07	19.12	30.87	36.82	37.14
Transformer ($T = 0.25$)	−3.92	2.75	5.73	14.01	8.46	22.76	35.17	39.37	32.40
Transformer ($T = 0.5$)	10.54	9.55	−1.27	−0.53	6.05	14.68	22.11	36.25	34.19
Transformer ($T = 1$, standard)	20.88	21.65	2.11	2.53	8.78	16.08	19.32	18.09	23.32
Transformer ($T = 2$)	45.80	30.01	26.12	5.64	21.63	17.18	25.97	20.89	23.92
Transformer ($T = 4$)	55.74	62.71	19.74	24.25	24.30	22.82	23.72	21.07	21.15

F.4 Making S6 more continuous

A stronger test of our claim is to intervene on S6 itself. Using the scalar α of Appendix F.3, we weaken the input dependence of Δ ($\alpha = 0.1$) or remove it altogether ($\alpha = 0$), and train otherwise unchanged S6 models on the six LRA tasks with the same hyperparameters. Table 1 shows that, as α decreases, accuracy improves markedly on the three more continuous tasks (Image, Pathfinder, and PathX), while it changes little, and if anything drops slightly, on the discrete language and symbolic tasks. The intervention also moves model continuity in the predicted direction: in the setting of Figure 3 with an input scaling factor of 32, the relative discretization error of S6 falls from 0.0103 at $\alpha = 1$ to 0.0054 at $\alpha = 0.1$ and to 0.0026 at $\alpha = 0$, nearly matching the 0.0025 of S4 (last row of Table 1). Meanwhile, the task-continuity scores computed from these encoders barely move (Table 7), so the gains come from a more continuous model rather than from tasks that merely look more continuous.

F.5 Stage-wise training across random seeds

The curves in Figure 6 come from a single run. To check their stability, we retrained the subsampled and the vanilla S4D models with three random seeds, as is standard in SSM work on LRA, and report the final test accuracies in Table 9. Except for ListOps, the subsampled training is stable across seeds, and the picture from the main text is preserved: vanilla training keeps a small edge on ListOps, the two are on par on Retrieval and Text, and the subsampled model wins on Image, Pathfinder, and PathX. The larger variation on ListOps is expected, since dropping an operation can completely change the meaning of an expression.

Table 9: Final test accuracy (in %, mean ± standard deviation over three random seeds) of the subsampled and the vanilla S4D models on the six LRA tasks.

Task	Subsampled S4D	Vanilla S4D
ListOps	59.83 ± 2.03	61.64 ± 0.92
Retrieval	91.21 ± 0.18	91.10 ± 0.21
Text	89.29 ± 0.32	89.20 ± 0.24
Image	91.02 ± 0.41	89.35 ± 0.31
Pathfinder	95.97 ± 0.43	95.65 ± 0.20
PathX	97.40 ± 0.19	96.63 ± 0.24

F.6 Irregularly sampled real-world time series

Finally, we step outside LRA and the synthetic systems, and compare S4D and S6 on three real-world forecasting benchmarks: Electricity, Weather, and ETTm1. Both models use six layers with hidden dimension 512, take 336 past observations as input, and directly forecast the next 96 steps; they are trained on the complete, regularly sampled data under the same protocol. At test time, we independently drop each past time index with probability $p \in \{0, 0.25, 0.5, 0.75\}$, using the same mask across channels and a single fixed masking seed, while leaving the 96-step target unchanged. During recurrent inference, the step size of each retained observation is multiplied by the number of grid points it has to cover: if the second, third, and fifth timestamps are dropped, the step of the first timestamp is tripled and that of the fourth is doubled. This is the inference-time counterpart of the subsampling in section 5. It tests how each model's discretization copes with an irregular time grid; it does not measure task continuity in the sense of μ. Table 10 reports the mean squared error in standardized units. S4D degrades gracefully as observations are removed, whereas S6 is far more sensitive, especially on Electricity and ETTm1. S4D does degrade noticeably on Electricity when only a quarter of the observations remain, which is expected: knowing that several hours have passed does not recover the measurements that were removed. The message is therefore two-fold. A model such as S4D can account for irregular time gaps directly, but timestamp awareness cannot replace missing values; the experiment also doubles as a simple diagnostic of how much a dataset relies on dense observations.

Table 10: Forecasting error (MSE in standardized units) of S4D and S6 on three real-world benchmarks when each past observation is dropped independently with probability p at test time. Both models are trained on the complete, regularly sampled data. S4D degrades gracefully, whereas S6 is far more sensitive to missing observations.

Dataset	Model	$p = 0$	$p = 0.25$	$p = 0.5$	$p = 0.75$
Electricity	S4D	0.2431	0.2538	0.2900	0.4322
	S6	0.2031	1.0565	0.9450	1.0361
Weather	S4D	0.2572	0.2587	0.2635	0.2677
	S6	0.2927	0.3610	0.3398	0.4172
ETTm1	S4D	0.4489	0.4516	0.4572	0.4898
	S6	0.4349	0.8618	0.8014	0.8783

G Limitations

Our work takes a first step toward formalizing continuity as an inductive bias in sequential modeling, but several directions remain open. On the theory side, our main analysis is developed for SSM-like models, where the underlying continuous-time formulation provides a natural starting point; extending comparable guarantees to broader model classes, especially Transformers and hybrid architectures, would be an important next step. On the empirical side, although we study several representative model families and tasks, we do not attempt an exhaustive survey of the rapidly growing space of sequential architectures, and the paper should therefore be viewed as identifying and isolating one important bias rather than as a complete guide for model selection. Finally, while our experiments show a clear empirical alignment between model continuity and task continuity, a deeper theoretical explanation for why more continuous models are better suited to more continuous tasks remains open. Two further caveats are worth stating. Our task-continuity score measures local proximity, so smoothness, locality, and low-frequency structure all raise it; conclusions drawn from the score should be read with this in mind, although its ordering of the tasks is stable across kernels, encoders, and lags (Appendix F.2). The stage-wise training strategy of section 5 reduces training time but not inference cost, and we view it as a demonstration of the continuity principle rather than a complete acceleration method. We believe each of these directions is substantial enough to support further dedicated study.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction state the main contributions of the paper accurately. In particular, they describe our three main directions: quantifying the continuity of SSM-like models, proposing a metric for the continuity of sequential datasets, and showing empirically that models with stronger continuity perform better on more continuous tasks. They also mention the training acceleration result as a practical consequence of continuity, without presenting it as the main focus of the paper. At the same time, the introduction makes clear that our theory is developed for SSM-like models and that our empirical study covers representative model families rather than the entire space of sequential architectures.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We provide a discussion of the limitations in Appendix G. We describe three main limitations: (1) our theoretical analysis is restricted to SSM-like models; (2) we do not aim to survey all sequential models, so the paper should not be read as a complete guide for model selection; and (3) while our experiments show a clear link between model continuity and task continuity, a deeper theoretical explanation of this phenomenon remains open.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.

- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

   Justification: All theoretical statements are stated in a self-contained way and contain all necessary assumptions, where cross-references to external equations are used for clarity. All proofs are given in Appendix B.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: The details of the experiments are given in Appendix C-E. Details that cannot be fully included in the appendices can be found in the associated code repository.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.

(c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: An anonymous code repository is submitted together with the paper. The repository contains all the necessary information and code to reproduce the experiments. Upon acceptance, a reproducibility section will be added to the paper, together with a link to a public GitHub repository.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Details of the hyperparameters are explained in Appendix C-E. We also clearly indicate a few cases where we inherit the hyperparameters from existing assets.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: The numerical experiments are accompanied by error bars, and all experiments on the dynamical systems have been repeated on multiple classes of dynamical systems to enhance the statistical significance. Some results on Long-Range Arena are not accompanied by error bars or standard deviations because they are collected from external works and do not come with statistical significance.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: The details are given in the caption of Figure 6 and appendices, where we describe the version of MATLAB used to produce the numerical experiments, and the computational resources used to train the subsampled models.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The authors have reviewed the NeurIPS Code of Ethics and the paper complies with it.

   Guidelines:

- The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [N/A]

    Justification: This paper is mainly theoretical and conceptual, and contains no large-scale experiments that might lead to any practical societal impact. There is no societal impact of the work performed.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The paper poses no such risks.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: The paper uses the Python implementations of S4 [20], S6 [18], and Transformers [53]. It also uses the Long-Range Arena benchmark collection [51]. All assets are properly cited.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: The paper does not release new assets.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.

- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: LLMs are only used for editing.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.