UnHiPPO: Uncertainty-aware Initialization for State Space Models

Marten Lienen [1 2] **Abdullah Saydemir** [1] **Stephan Günnemann** [1 2]

Abstract

State space models are emerging as a dominant model class for sequence problems with many relying on the HiPPO framework to initialize their dynamics. However, HiPPO fundamentally assumes data to be noise-free; an assumption often violated in practice. We extend the HiPPO theory with measurement noise and derive an uncertainty-aware initialization for state space model dynamics. In our analysis, we interpret HiPPO as a linear stochastic control problem where the data enters as a noise-free control signal. We then reformulate the problem so that the data become noisy outputs of a latent system and arrive at an alternative dynamics initialization that infers the posterior of this latent system from the data without increasing runtime. Our experiments show that our initialization improves the resistance of state-space models to noise both at training and inference time.

1. Introduction

In 2019, Voelker et al. proposed a novel memory cell, the Legendre Memory Unit (LMU), for recurrent neural networks. Their cell keeps a continuously updated representation of input in a sliding window in terms of orthogonal polynomials. After Gu et al. (2020) generalized LMUs to variable length input windows, they were further extended as state space models (SSMs) by Gu et al. (2021) and applied across a multitude of domains. Their ability to capture temporal dependencies over long sequences made them a great fit for natural language processing (Mehta et al., 2022), time series analysis (Patro & Agneeswaran, 2024b), speech generation (Goel et al., 2022) and more (Patro & Agneeswaran, 2024a).

A significant contribution to the success of SSMs is the high-order polynomial projection operator (HiPPO) initialization for their dynamics introduced by Gu et al. (2020). The HiPPO dynamics compress a sequence into a finite-dimensional state that represents the projection of the sequence onto a basis of Legendre polynomials. Importantly, maintaining this state requires only the current state and the next value in the sequence and is thus independent of the sequence length. This enables the construction of SSMs that efficiently capture long-range dependencies with linear computational complexity. Such SSMs are well-suited for deep learning applications where sequence length can be substantial, such as language (Gu & Dao, 2023) or video modeling (Park et al., 2025).

However, the original HiPPO framework assumes that the observed data is noise-free – an assumption that is easily violated. In many practical scenarios, measurements are contaminated with noise due to sensor imperfections, environmental factors, or inherent variability in the data-generating process. Thus, this noise-free assumption limits the applicability of HiPPO-initialized SSMs, as they may perform suboptimally when exposed to noisy observations commonly encountered in real-world applications.

We address this limitation by extending the HiPPO framework to explicitly account for measurement noise. By reinterpreting HiPPO as a linear stochastic control problem where data emerges as noisy observations of a latent system, we derive the uncertainty-aware HiPPO (UnHiPPO) initialization for SSMs. With UnHiPPO, an SSM implicitly performs posterior inference in a linear dynamical system, making it robust against noise in the data without increasing computational complexity or changes to the model structure.

Our **contributions** can be summarized as follows:

- We provide a thorough description and derivation of HiPPO and its application to SSM initialization in Sections A to 3.
- In Section 4, we analyze HiPPO with respect to its noise robustness using the framework of linear stochastic control theory. We modify HiPPO based on this analysis so that it performs implicit posterior inference under the assumption of observation noise and propose a regularization technique to make this numerically stable.

[1]Department of Computer Science, Technical University of Munich [2]Munich Data Science Institute, Technical University of Munich. Correspondence to: Marten Lienen <m.lienen@tum.de>.

Proceedings of the 42^{nd} International Conference on Machine Learning, Vancouver, Canada. PMLR 267, 2025.

- In Section 5, we present our uncertainty-aware extension of HiPPO and show how to apply it as an initialization in the Linear State Space Layer (LSSL) model (Gu et al., 2021). Furthermore, we analyze the effect of applying the time-varying UnHiPPO dynamics in a time-invariant way and why this can be interpreted as operating a model on different time scales.
- Our experiments in Section 6 demonstrate how the UnHiPPO initialization improves the robustness of SSMs against noise using the example of LSSL.

Find our implementation at cs.cit.tum.de/daml/unhippo.

Notation We use 0-based indexing for the coefficient vectors and transition matrices to stay consistent with the natural numbering of polynomial basis functions where g_i is of degree i. Furthermore, $t \in \mathbb{R}^+$ denotes the current point in time and $\tau \in [0, t]$ another point in the signal up to time t. x is a point in the subset of $\mathbb{R}$ where the polynomials under consideration form an orthogonal basis, e.g. $x \in [-1, 1]$ for Legendre polynomials. $[n]$ refers to the set of integers $1, \dots, n$.

2. Legendre Polynomials

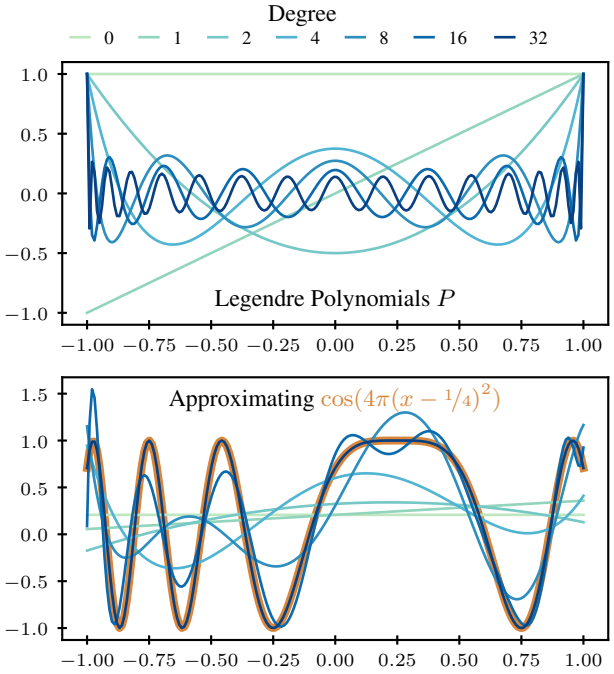

Figure 1. Function approximation with Legendre polynomials.

The Legendre polynomials $P_i : \mathbb{R} \to \mathbb{R}, i \in \mathbb{N}_0$ are a set of polynomials of increasing degree i orthogonal with respect to the L^2 inner product $\langle f, g \rangle = \int_{-1}^{1} f(x)\, g(x)\, \mathrm{d}x$. Together, they form a basis of the space of L^2-integrable functions on $[-1, 1]$ and let us represent any such function f as

$$\hat{f}(x) = \sum_{i=0}^{D} \frac{\langle f, P_i \rangle}{\|P_i\|^2} P_i(x). \tag{1}$$

$\hat{f}$ is the best possible approximation to f with respect to the distance induced by $\langle \cdot, \cdot \rangle$ in terms of polynomials up to degree D and can be seen as the projection of f onto the space spanned by the basis functions $P_0, \dots, P_D$. Fig. 1 illustrates these concepts.

In this sense, the vector $c_i = \langle f, P_i \rangle, i = 0, \dots, D$ is the best possible compression of f in the L^2 space into $D + 1$ numbers as it lets us reconstruct f as $\hat{f}$. For more background on polynomial approximation of signals in general and Legendre polynomials in particular, we refer the reader to (Vetterli et al., 2014; Arfken & Weber, 2008).

3. HiPPO

Let's say, we observe a scalar signal $f(\tau) : \mathbb{R}^+ \to \mathbb{R}$ and want to process it in an online fashion with a downstream model $h : \mathbb{R}^N \to \mathcal{X}$, for example time series classification or a running forecast. Since h processes fixed-length vectors, we need a vector representation of length N of the signal history $f_{\leq t} \coloneqq f|_{[0,t]}$ up to the current time t that is expressive and informative.

Gu et al. (2020) propose the HiPPO that represents $f_{\leq t}$ as the coefficient vector $\boldsymbol{c}_t$ of the projection $\hat{f}_{\leq t}$ of $f_{\leq t}$ onto a basis of slightly modified Legendre polynomials g_i up to degree $D = N - 1$. While this representation is expressive, its computation via Eq. (1) would require the complete signal $f_{\leq t}$ at each point t. This would make both runtime and memory requirements scale with t, rendering this representation inefficient for long time series.

Crucially, Gu et al. (2020) show how the coefficients $\boldsymbol{c}_{t_{k+1}}$ can be computed approximately from coefficients $\boldsymbol{c}_{t_k}$ at an earlier time t_k and the value of the signal f at time t_{k+1} with a linear update step. This means that we only need to remember the current coefficients $\boldsymbol{c}_{t_k}$ instead of the complete signal history $f_{\leq t}$ to update the compressed representation of the signal when a new observation $y_{t_{k+1}} = f(t_{k+1})$ comes in. As a result, HiPPO gives us N expressive features describing $f_{\leq t}$ without having to store the complete history. In fact, the memory requirements of HiPPO are $\mathcal{O}(N)$ and thus independent of the length of the sequence.

In the following, we will present a condensed description of HiPPO to introduce the formal background for our extension in Sections 4 and 5. See Section A for a complete and expository derivation.

Shifting and scaling As a first step, the Legendre polynomials have to be shifted and scaled to form an or-

thonormal basis on the interval $[0, t]$. For that, Gu et al. (2020) choose the time-dependent inner product $\langle f, g\rangle_t = {}^1\!/_t \int_0^t f(x)\, g(x)\, \mathrm{d}x$, under which the basis polynomials

$$g_{t,i}(\tau) = \sqrt{2i+1}\, P_i(\phi_t(\tau)) \tag{2}$$

are orthonormal. $\phi_t(\tau) = {}^{2\tau}\!/_t - 1$ maps $[0, t]$ linearly onto $[-1, 1]$. This gives us the vector representation

$$c_{t,i} = \langle f_{\leq t}, g_{t,i}\rangle_t \tag{3}$$

of the complete history of the signal f until time t.

Online update To maintain the coefficient vector $\boldsymbol{c}_t$ as time passes and we observe more of the signal f, Gu et al. (2020) take the time derivative of Eq. (3) to find the vector ordinary differential equation (ODE)

$$\frac{\mathrm{d}\boldsymbol{c}_t}{\mathrm{d}t} = -\frac{1}{t}\boldsymbol{A}_{\mathrm{H}}\boldsymbol{c}_t + \frac{1}{t}\boldsymbol{B}_{\mathrm{H}} f(t), \tag{4}$$

which describes the evolution of the coefficient vector $\boldsymbol{c}_t$ in continuous time under the influence of the signal $f(t)$. The HiPPO matrix $\boldsymbol{A}_{\mathrm{H}}$ and vector $\boldsymbol{B}_{\mathrm{H}}$ are given by

$$A_{\mathrm{H,ij}} = \begin{cases} \sqrt{2i+1}\sqrt{2j+1} & \text{if } j < i, \\ i+1 & \text{if } j = i, \\ 0 & \text{if } j > i, \end{cases} \tag{5}$$

$$\text{and} \quad B_{\mathrm{H,i}} = \sqrt{2i+1}.$$

Discretization Applying the update to discrete data observed at times $t_1, t_2, \ldots$ requires a discrete analog of Eq. (4). To this end, Gu et al. (2020) discretize the equation into the recurrence

$$\boldsymbol{c}_{t_{k+1}} = \bar{\boldsymbol{A}}_{\mathrm{H}}\boldsymbol{c}_{t_k} + \bar{\boldsymbol{B}}_{\mathrm{H}} y_{t_{k+1}} \tag{6}$$

as described in Section A.3. Eq. (6) can then serve as the basis for a recurrent neural network layer, similar to GRU (Cho et al., 2014) or LSTM (Hochreiter & Schmidhuber, 1997).

See Section A for an explanatory derivation of the results in this section.

4. From Control to Inference

To understand the behavior of HiPPO under measurement noise, we will analyze it as a linear stochastic control problem (Aström, 1970). In this theory, a continuous-discrete linear dynamical system (LDS) with scalar control and observations is described as

$$\mathrm{d}\boldsymbol{x}_t = \big(\boldsymbol{\Phi}\boldsymbol{x}_t + \boldsymbol{\Gamma}u_t\big)\mathrm{d}t + \mathrm{d}\boldsymbol{\beta} \tag{7}$$

$$y_{t_k} = \boldsymbol{\Psi}\boldsymbol{x}_{t_k} + \varepsilon_{t_k}. \tag{8}$$

Such a system has a latent state $\boldsymbol{x}_t$ that we can influence with a scalar control signal u_t via its linear stochastic dynamics in Eq. (7). At discrete times t_k, we take noisy measurements y_{t_k} of the system in Eq. (8). $\mathrm{d}\boldsymbol{\beta}$ is white noise, ε_{t_k} is independent Gaussian noise and $\boldsymbol{\Phi}$, $\boldsymbol{\Gamma}$ and $\boldsymbol{\Psi}$ are parameter matrices. Note that Eqs. (7) and (8) include exactly two sources of noise: in the state dynamics and during the measurements.

If we compare the HiPPO dynamics in Eq. (4) to the above system, we see that $\boldsymbol{c}_t$ takes the role of the system state $\boldsymbol{x}_t$. In contrast to what one might expect, the observed signal $f(t)$ does not correspond to the observations y_{t_k} in HiPPO and, instead, it corresponds to the control signal u_t, which is assumed to be noise-free in linear stochastic control theory. This means that HiPPO fundamentally assumes that there is no measurement noise on the signal $f(t)$ nor any noise in the system dynamics themselves and that $\boldsymbol{c}_t$ is a deterministic function of $f_{\leq t}$.

However, in many applications, these assumptions do not hold. This includes basically all time series modeling where the data is derived from a physical process such as a temperature, wind speed or power output, but also extends to exactly measurable but inherently noisy data such as user interactions and financial transaction logs.

Guarding the system against noise requires us to take it explicitly into account in the model. Consequently, we design a continuous-discrete LDS based on the HiPPO dynamics in Eq. (4) but with $f(t)$ modeled via noisy observations instead of a noise-free control signal. In the end, $f(t)$ will not appear directly in the model at all. Instead, we will infer the posterior distribution $\mathrm{p}(\boldsymbol{c}_k \mid y_{t_{1:k}})$, filtering out the effect of noise in the measurements and transitions.

To begin, we need to specify a model for $y_{t_k} = f(t_k) + \varepsilon$ linear in c_{t_k}. Since c_{t_k} represents the approximate signal $\hat{f}_{\leq t_k}$ until t_k, we predict $f(t_k)$ with

$$\hat{f}_{\leq t_k}(t_k) = \sum_{i=0}^{N} c_{t_k,i}\, g_{t_k,i}(t_k) = \boldsymbol{B}_{\mathrm{H}}^{\mathsf{T}}\boldsymbol{c}_{t_k}, \tag{9}$$

and model

$$y_{t_k} = \boldsymbol{B}_{\mathrm{H}}^{\mathsf{T}}\boldsymbol{c}_{t_k} + \varepsilon_{t_k} \quad \text{where} \quad B_{\mathrm{H,i}} = \sqrt{2i+1}. \tag{10}$$

Secondly, we need to make the HiPPO dynamics independent of $f(t)$ while keeping them linear. For that, we substitute Eq. (9) for $f(t)$ in Eq. (4) and get

$$\frac{\mathrm{d}\boldsymbol{c}_t}{\mathrm{d}t} = \frac{1}{t}\big(\boldsymbol{B}_{\mathrm{H}}\boldsymbol{B}_{\mathrm{H}}^{\mathsf{T}} - \boldsymbol{A}_{\mathrm{H}}\big)\boldsymbol{c}_t, \tag{11}$$

which we can simplify with the identity $\boldsymbol{B}_{\mathrm{H}}\boldsymbol{B}_{\mathrm{H}}^{\mathsf{T}} - \boldsymbol{A}_{\mathrm{H}} = \boldsymbol{A}_{\mathrm{H}}^{\mathsf{T}} - \boldsymbol{I}$ which follows directly from the definitions in Eq. (5). Since the HiPPO dynamics are exact, the dynamics

in Eq. (11) give us the exact evolution of $\boldsymbol{c}_t$ if the true signal $f_{\leq t}$ is just the polynomial $\hat{f}_{\leq t}$.

By combining the model for y_{t_k} and the data-free dynamics, we get the first version of our noise-resistant LDS.

$$\begin{aligned} \mathrm{d}\boldsymbol{c}_t &= \frac{1}{t}\left(\boldsymbol{A}_{\mathrm{H}}^{\mathsf{T}} - \boldsymbol{I}\right)\boldsymbol{c}_t \mathrm{d}t + \mathrm{d}\boldsymbol{\beta} \\ y_{t_k} &= \boldsymbol{B}_{\mathrm{H}}^{\mathsf{T}} \boldsymbol{c}_{t_k} + \varepsilon_{t_k} \end{aligned} \tag{12}$$

4.1. Extrapolation

Inferring $\boldsymbol{c}_t$ from data with Eq. (12) will come down to two steps. First, we solve Eq. (11) forward in time from t_k to t_{k+1} to get coefficients $\boldsymbol{c}_{t_{k+1}}$ that represent the extrapolation of $\hat{f}_{\leq t_k}$ from t_k to t_{k+1}. Second, we update the forecast $\boldsymbol{c}_{t_{k+1}}$ with the observed data $y_{t_{k+1}}$. In contrast, HiPPO uses the data directly in the forward solving of Eq. (4).

A complication with our first step is that Legendre polynomials – and by extension $\hat{f}_{\leq t_k}$ – are terribly unfit for extrapolation. While each $g_{t,i}$ is well behaved within $[0, t]$, they diverge rapidly towards $\pm\infty$ outside of it – like an i-th degree polynomial to be exact.

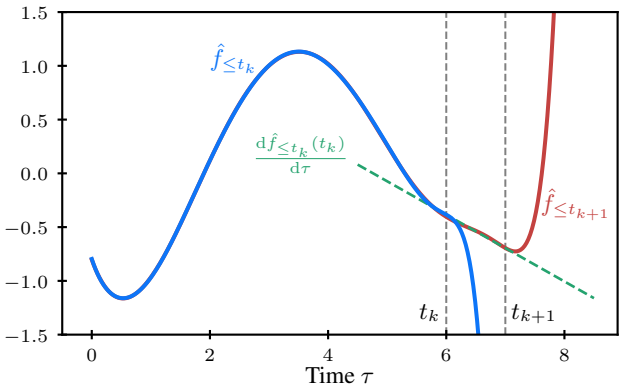

Figure 2. The Legendre polynomial $\hat{f}_{\leq t_k}$ diverges rapidly beyond its domain $[0, t_k]$. If we extrapolate $\hat{f}_{\leq t_k}$ with Eq. (11), $\hat{f}_{\leq t_{k+1}}$ will equal $\hat{f}_{\leq t_k}$ but on the extended domain $[0, t_{k+1}]$. By regularizing the dynamics with the conditions in Eq. (13), we can get the $\hat{f}_{\leq t_{k+1}}$ shown here, that extrapolates $\hat{f}_{\leq t_k}$ linearly until t_{k+1}.

This is a problem from two perspectives. First, when we model $f_{\leq t}$ with, for example, $N = 256$ coefficients, $\hat{f}_{\leq t_k}$ diverges like a degree 255 polynomial outside of $[0, t_k]$ and any predicted $y_{t_{k+1}}$ will almost surely be a horrible prediction of the true $f(t_{k+1})$. Second, numerical computations with a forecast of such magnitude will produce enough floating point cancellation to invalidate any results.

To reduce the order of divergence, we propose to regularize the system dynamics. In particular, we are looking for dynamics that retain the shape of $\hat{f}$ as much as possible up to t_k but evolve $\boldsymbol{c}_t$ such that the slope of $\hat{f}$ remains roughly constant from t_k onwards. Fig. 2 visualizes this goal. By retaining the shape of $\hat{f}$, we preserve the information embedded in $\boldsymbol{c}_t$.

Formally, we can express our desired behavior of $\hat{f}_{\leq t}$ as

$$\frac{\mathrm{d}\hat{f}_{\leq t}(t)}{\mathrm{d}t} = \frac{\mathrm{d}\hat{f}_{\leq t}(t)}{\mathrm{d}\tau} \quad \text{and} \quad \frac{\mathrm{d}}{\mathrm{d}t}\frac{\mathrm{d}\hat{f}_{\leq t}(t)}{\mathrm{d}\tau} = 0. \tag{13}$$

In words, the first condition demands that the value at t of the reconstruction $\hat{f}_{\leq t}$, i.e. the current value of our signal, follow the gradient of $\hat{f}_{\leq t}$ as we extend the domain and predict into the future. If you compare to Fig. 2, we want that the rightmost value of $\hat{f}_{\leq t}$ follows the dashed line as we increase t. The second condition says that this gradient shall not change. Remember that the parameter of $\hat{f}_{\leq t}$ is τ, so the two derivatives in the conditions refer to the subscript and the parameter, respectively.

With Eq. (9), we see immediately that

$$\frac{\mathrm{d}\hat{f}_{\leq t}(t)}{\mathrm{d}t} = \boldsymbol{B}_{\mathrm{H}}^{\mathsf{T}} \frac{\mathrm{d}\boldsymbol{c}_t}{\mathrm{d}t}. \tag{14}$$

The τ-derivative of $\hat{f}_{\leq t}(\tau)$ at $\tau = t$ becomes

$$\frac{\mathrm{d}\hat{f}_{\leq t}(t)}{\mathrm{d}\tau} = \sum_{i=0}^{N} c_{t,i} \frac{\mathrm{d}g_{t,i}(t)}{\mathrm{d}\tau} = \frac{2}{t}\boldsymbol{Q}^{\mathsf{T}}\boldsymbol{c}_t \tag{15}$$

where $Q_i = \sqrt{2i+1}\frac{\mathrm{d}P_i(1)}{\mathrm{d}x} = \sqrt{2i+1}\frac{i(i+1)}{2}$ which can be derived from Eq. (2) and (Arfken & Weber, 2008, Eq. (12.24))

$$\frac{\mathrm{d}P_i(x)}{\mathrm{d}x} = iP_{i-1}(x) + x\frac{\mathrm{d}P_{i-1}(x)}{\mathrm{d}x}. \tag{16}$$

Now, we can combine Eqs. (11) and (13) into the condition

$$\begin{pmatrix} \boldsymbol{I} \\ \boldsymbol{B}_{\mathrm{H}}^{\mathsf{T}} \\ \boldsymbol{Q}^{\mathsf{T}} \end{pmatrix} \frac{\mathrm{d}\boldsymbol{c}_t}{\mathrm{d}t} = \frac{1}{t}\begin{pmatrix} \boldsymbol{A}_{\mathrm{H}}^{\mathsf{T}} - \boldsymbol{I} \\ 2\boldsymbol{Q}^{\mathsf{T}} \\ \boldsymbol{Q}^{\mathsf{T}} \end{pmatrix} \boldsymbol{c}_t. \tag{17}$$

Since this is overdetermined, we cannot solve for $\frac{\mathrm{d}\boldsymbol{c}_t}{\mathrm{d}t}$ exactly, but we can find an approximate solution with the pseudo-inverse denoted by $\dagger$,

$$\frac{\mathrm{d}\boldsymbol{c}_t}{\mathrm{d}t} = \frac{1}{t}\underbrace{\begin{pmatrix} \boldsymbol{I} \\ \boldsymbol{B}_{\mathrm{H}}^{\mathsf{T}} \\ \boldsymbol{Q}^{\mathsf{T}} \end{pmatrix}^{\dagger}\begin{pmatrix} \boldsymbol{A}_{\mathrm{H}}^{\mathsf{T}} - \boldsymbol{I} \\ 2\boldsymbol{Q}^{\mathsf{T}} \\ \boldsymbol{Q}^{\mathsf{T}} \end{pmatrix}}_{=:\boldsymbol{A}_{\mathrm{R}}} \boldsymbol{c}_t. \tag{18}$$

We call $\boldsymbol{A}_{\mathrm{R}}$ the *regularized HiPPO matrix* and define the regularized LDS

$$\mathrm{d}\boldsymbol{c}_t = \frac{1}{t}\boldsymbol{A}_{\mathrm{R}}\boldsymbol{c}_t \mathrm{d}t + \mathrm{d}\boldsymbol{\beta}, \qquad y_{t_k} = \boldsymbol{B}_{\mathrm{H}}^{\mathsf{T}}\boldsymbol{c}_{t_k} + \varepsilon_{t_k}. \tag{19}$$

4.2. Discretization

At this point, we move from the semi-discretized to the fully discretized case. As described by Särkkä & Solin (2019, Section 6), the discrete-time model equivalent to Eq. (19) is

$$\begin{aligned} \boldsymbol{c}_{k+1} &= \bar{\boldsymbol{A}}_{\mathrm{R,k+1}} \boldsymbol{c}_k + \boldsymbol{q}_k \\ y_k &= \boldsymbol{B}_{\mathrm{H}}^{\mathsf{T}} \boldsymbol{c}_k + \varepsilon_k \end{aligned} \tag{20}$$

where $\bar{\boldsymbol{A}}_{\mathrm{R,k+1}}$ is the transition matrix from t_k to t_{k+1}, i.e.

$$\bar{\boldsymbol{A}}_{\mathrm{R,k+1}} = \boldsymbol{I} + \int_{t_k}^{t_{k+1}} \frac{1}{\tau} \boldsymbol{A}_{\mathrm{R}} \bar{\boldsymbol{A}}_{\mathrm{R,k}} \, \mathrm{d}\tau. \tag{21}$$

$\boldsymbol{q}_k$ and ε_k are zero-mean Gaussian noise variables with covariance $\boldsymbol{\Sigma}$ and σ^2, respectively, as hyperparameters.

In contrast to HiPPO, the dynamics in Eq. (18) do not depend on the observations. This means that we can evaluate them, and in particular $1/t \, \boldsymbol{A}_{\mathrm{R}}$, at any t, not just the t_k where we have observed data. Therefore, we can use any ODE solver to approximate the transition matrix via Eq. (21).

We can, for example, approximate $\bar{\boldsymbol{A}}_{\mathrm{R,k+1}}$ with any integration rule considered by Gu et al. (2021), e.g. a forward Euler step $\boldsymbol{I} + \frac{\Delta t}{t_k} \boldsymbol{A}_{\mathrm{R}}$, a backward Euler step $\left(\boldsymbol{I} - \frac{\Delta t}{t_{k+1}} \boldsymbol{A}_{\mathrm{R}}\right)^{-1}$ or the trapezoidal rule $\left(\boldsymbol{I} - \frac{\Delta t}{2t_{k+1}} \boldsymbol{A}_{\mathrm{R}}\right)^{-1} \left(\boldsymbol{I} + \frac{\Delta t}{2t_k} \boldsymbol{A}_{\mathrm{R}}\right)$ where $\Delta t = t_{k+1} - t_k$. However, in contrast to HiPPO, there exists an exact closed-form solution $\exp(\log(t_{k+1}/t_k) \, \boldsymbol{A}_{\mathrm{R}})$ to Eq. (21), which we use instead of approximations. We show in Section B that it produces the most stable linear recurrence. Note that it is a matrix exponential, not a componentwise application.

4.3. Posterior Distribution

Next, we connect the model specified in Eq. (20) to the data. Since both dynamics and observations are linear, we can compute the posterior $\mathrm{p}(\boldsymbol{c}_k \mid y_{1:k}) = \mathcal{N}(\boldsymbol{m}_k, \boldsymbol{P}_k)$ in closed form via the Kalman filter (Kalman, 1960; Särkkä & Solin, 2019).

For that, we put a standard Gaussian prior $\mathcal{N}(\boldsymbol{m}_0, \boldsymbol{P}_0)$ on $\boldsymbol{c}_0$, i.e. $\boldsymbol{m}_0 = \boldsymbol{0}$ and $\boldsymbol{P}_0 = \boldsymbol{I}$. Then, we can compute $\mathrm{p}(\boldsymbol{c}_k \mid y_{1:k}) = \mathcal{N}(\boldsymbol{m}_k, \boldsymbol{P}_k)$ through the following recurrence. First, we roll out the dynamics for one step to extend $\hat{f}_{\leq t_k}$ to $\hat{f}_{\leq t_{k+1}}$.

$$\begin{aligned} \boldsymbol{m}_k^- &= \bar{\boldsymbol{A}}_{\mathrm{R,k}} \boldsymbol{m}_{k-1} \\ \boldsymbol{P}_k^- &= \bar{\boldsymbol{A}}_{\mathrm{R,k}} \boldsymbol{P}_{k-1} \bar{\boldsymbol{A}}_{\mathrm{R,k}}^{\mathsf{T}} + \boldsymbol{\Sigma} \end{aligned} \tag{22}$$

Then, we compare the prediction to the observed data y_k and update the posterior parameters accordingly.

$$\begin{aligned} v_k &= y_k - \boldsymbol{B}_{\mathrm{H}}^{\mathsf{T}} \boldsymbol{m}_k^- \\ s_k &= \boldsymbol{B}_{\mathrm{H}}^{\mathsf{T}} \boldsymbol{P}_k^- \boldsymbol{B}_{\mathrm{H}} + \sigma^2 \\ \boldsymbol{K}_k &= {}^1/_{s_k} \, \boldsymbol{P}_k^- \boldsymbol{B}_{\mathrm{H}} \\ \boldsymbol{m}_k &= \boldsymbol{m}_k^- + \boldsymbol{K}_k v_k \\ \boldsymbol{P}_k &= \boldsymbol{P}_k^- - s_k \boldsymbol{K}_k \boldsymbol{K}_k^{\mathsf{T}} \end{aligned} \tag{23}$$

Since there is no 0-th data point and thus no t_0, we set $t_0 = t_1$ to compute $\bar{\boldsymbol{A}}_{\mathrm{R},1}$, giving $\bar{\boldsymbol{A}}_{\mathrm{R},1} = \boldsymbol{I}$.

Numerical Stability Direct application of the Kalman filter can be numerically challenging, because the covariance matrix $\boldsymbol{P}_k$ often has some eigenvalues close to zero. The smallest eigenvalues can then flip into the negative due to floating point cancellation in the matrix difference in Eq. (23), losing the positive semi-definiteness of $\boldsymbol{P}_k$. Furthermore, because floating point addition and multiplication are non-associative (Goldberg, 1991), $\boldsymbol{P}_k$ can also lose its symmetry, making the filter diverge. In our experiments, we solved the former issue by computing the Kalman filter in double precision and the latter with symmetrization $\boldsymbol{P}_k \coloneqq (\boldsymbol{P}_k + \boldsymbol{P}_k^{\mathsf{T}})/2$ after each update.

5. Uncertainty-aware Initialization

Upon closer inspection of the posterior distribution of $\boldsymbol{c}_k$, we notice two things. First, the posterior covariance contains no information about the data as a consequence of the assumptions of the Kalman filter. Second, we can combine the whole procedure in Eqs. (22) and (23) into a single linear equation for the posterior mean $\boldsymbol{m}_k$.

$$\boldsymbol{m}_k = \underbrace{\left(\boldsymbol{I} - \boldsymbol{K}_k \boldsymbol{B}_{\mathrm{H}}^{\mathsf{T}}\right) \bar{\boldsymbol{A}}_{\mathrm{R,k}}}_{\bar{\boldsymbol{A}}_{\mathrm{U,k}}} \boldsymbol{m}_{k-1} + \underbrace{\boldsymbol{K}_k}_{\bar{\boldsymbol{B}}_{\mathrm{U,k}}} y_k \tag{24}$$

We call $\bar{\boldsymbol{A}}_{\mathrm{U,k}}$ and $\bar{\boldsymbol{B}}_{\mathrm{U,k}}$ the *uncertainty-aware HiPPO* (UnHiPPO) *matrix* and *vector*, respectively.

Eq. (24) is an uncertainty-aware equivalent to the discretized HiPPO dynamics in Eq. (6). Fig. 3 compares the two in the face of noise. It shows a random function sampled from a Gaussian process as the ground-truth signal and noisy observations derived from the signal by adding independent Gaussian noise. On top of the data, we see the reconstructions $\hat{f}_{\leq t_k}$ created from the coefficients gathered from the HiPPO and UnHiPPO dynamics, respectively. This lets us visually compare the effect of the observation noise on the features that these dynamics extract from the data. We see that the noise introduces a lot of spurious high-frequency signal in the HiPPO representation of the data, while the UnHiPPO dynamics filter out the majority of the noise and extract a close approximation of the ground-truth signal.

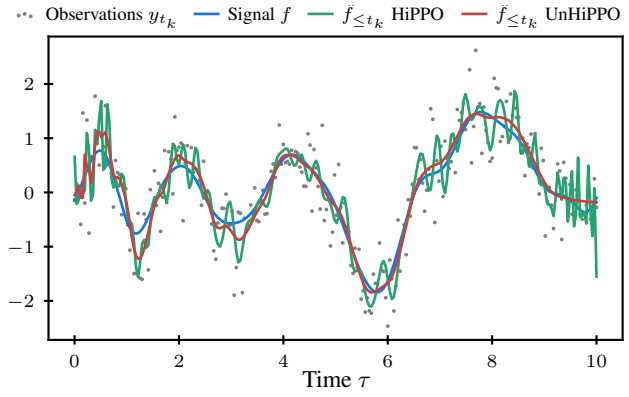

Figure 3. The uncertainty-aware HiPPO dynamics filter out the majority of the noise.

The strength of the denoising is controlled by the σ^2 hyperparameter through its influence on the so-called innovation covariance s_k in the Kalman update step in Eq. (23). Fig. 4 visualizes the effect spectrum of σ^2. Too small values mean that the signal is modeled closely including noise, while too large values make the dynamics ignore the data. For optimal filtering, σ^2 needs to be adapted to the data. Note the surprisingly large scale of σ^2 in Fig. 4. This is necessary because σ^2 needs to overcome the $\boldsymbol{B}_{\mathrm{H}}^{\mathsf{T}}\boldsymbol{P}_k^{-}\boldsymbol{B}_{\mathrm{H}}$ term in s_k, which is large because of the magnitude of entries of the HiPPO matrix in Eq. (5). An unfortunate side effect is that σ^2 cannot be interpreted as the noise variance of the data directly.

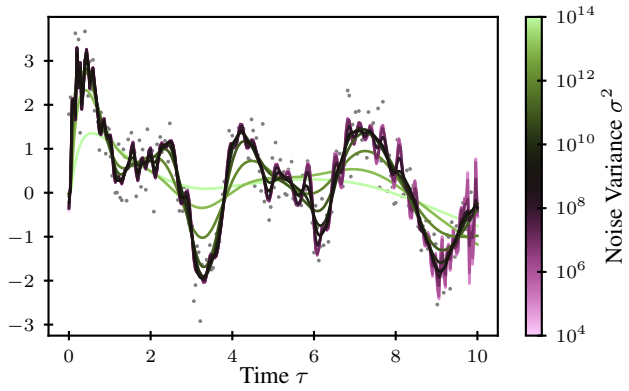

Figure 4. The σ^2 hyperparameter controls the level of filtering.

For the transition uncertainty, we always choose $\boldsymbol{\Sigma} = \boldsymbol{I}$. In our experiments, it had no appreciable effect except for instability if it was very small.

5.1. Uncertainty-aware Linear State Space Layer

The LSSL (Gu et al., 2021) is a simple sequence model that maps a scalar input sequence $u_k, k \in [n]$ to a scalar output sequence y_k via a latent state $\boldsymbol{c}_k \in \mathbb{R}^N$ and the recurrence

$$\begin{aligned} \boldsymbol{c}_k &= \boldsymbol{A}\boldsymbol{c}_{k-1} + \boldsymbol{B}u_k \\ y_k &= \boldsymbol{C}\boldsymbol{c}_k + \boldsymbol{D}u_k \end{aligned} \tag{25}$$

where $\boldsymbol{A} \in \mathbb{R}^{N\times N}$, $\boldsymbol{B} \in \mathbb{R}^N$, $\boldsymbol{C} \in \mathbb{R}^{1\times N}$ and $\boldsymbol{D} \in \mathbb{R}$ are parameters.

Gu et al. (2021) create a building block for neural sequence models from this by stacking H independent copies of Eq. (25) for an H-dimensional input, i.e. the output of the previous layer. To improve the expressiveness of the system, they introduce M latent channels by increasing the dimension of the output $\boldsymbol{y}_k$ to M. Since simple stacking means that each feature of the input $\boldsymbol{u}_k$ is processed independently, they also introduce a position-wise map to mix the channels. This map consists of a GELU nonlinearity followed by a linear layer that maps $\boldsymbol{y}_k \in \mathbb{R}^{H\times M}$ to the final output of dimensionality H, i.e. the weight matrix of the linear layer has dimensions $(H \cdot M) \times H$. The stacked parameters become $\boldsymbol{A} \in \mathbb{R}^{H\times N\times N}$, $\boldsymbol{B} \in \mathbb{R}^{H\times N}$, $\boldsymbol{C} \in \mathbb{R}^{H\times M\times N}$ and $\boldsymbol{D} \in \mathbb{R}^{H\times M}$.

The initialization of $\boldsymbol{A}$ and $\boldsymbol{B}$ is essential as Gu et al. (2021) have shown and the HiPPO initialization produced much stronger results than a random initialization. They discretize the HiPPO dynamics to

$$\boldsymbol{c}_k \approx \bar{\boldsymbol{A}}_{\mathrm{H},k}\boldsymbol{c}_{k-1} + \bar{\boldsymbol{B}}_{\mathrm{H},k}y_k \tag{26}$$

where

$$\bar{\boldsymbol{A}}_{\mathrm{H},k} = \left(\boldsymbol{I} + \frac{\Delta t}{2t_k}\boldsymbol{A}_{\mathrm{H}}\right)^{-1}\left(\boldsymbol{I} - \frac{\Delta t}{2t_k}\boldsymbol{A}_{\mathrm{H}}\right) \tag{27}$$

and

$$\bar{\boldsymbol{B}}_{\mathrm{H},k} = \left(\boldsymbol{I} + \frac{\Delta t}{2t_k}\boldsymbol{A}_{\mathrm{H}}\right)^{-1}\frac{\Delta t}{t_k}\boldsymbol{B}_{\mathrm{H}}. \tag{28}$$

See Section A.3 for a derivation of this discretization. Then, they initialize $\boldsymbol{A}_i = \bar{\boldsymbol{A}}_{\mathrm{H},i}$ and $\boldsymbol{B}_i = \bar{\boldsymbol{B}}_{\mathrm{H},i}$ where they set $\Delta t = 1$ and vary t_k log-uniformly between $t_{\min}$ for $i = 1$ and $t_{\max}$ for $i = H$.

We adapt this into the uncertainty-aware LSSL (UnLSSL) by initializing

$$\boldsymbol{A}_i = \bar{\boldsymbol{A}}_{\mathrm{U},\lfloor t\rfloor} \qquad \text{and} \qquad \boldsymbol{B}_i = \bar{\boldsymbol{B}}_{\mathrm{U},\lfloor t\rfloor} \tag{29}$$

where t varies in the same way as for LSSL. One difference to LSSL is in how these initializations are computed. In contrast to LSSL where we can get the discretized dynamics at any t directly, for UnLSSL we compute them for all integer steps $t \in [t_{\max}]$ and then select a subset. In theory, we could jump to any t directly in the Kalman update in Eq. (23), but that would increase the uncertainty $\boldsymbol{P}_k^{-}$ as if there was no data before t, changing the dynamics. Instead, we also

compute all intermediate steps, which mirrors the more realistic setting where we also observe data at $1, 2, \ldots, t-1$. Note that this only happens once for initialization and has negligible runtime cost.

Gu et al. (2021) improve the runtime of LSSL by rewriting Eq. (26) from a recurrence to a convolution with a Krylov kernel that can be constructed in $O(\log n)$ time. Since UnLSSL is just a different initialization for the same dynamics, the same optimization applies. Note that the Krylov kernel for UnLSSL needs to be constructed in double precision, though it can be cast to single precision before the convolution.

$\boldsymbol{A}_i$ and $\boldsymbol{B}_i$ can be fine-tuned with gradient descent during model training. However, Gu et al. (2021) have shown that training $\boldsymbol{A}_i$ and $\boldsymbol{B}_i$ has only a minor effect on LSSL performance in sequence classification, but prevents caching the Krylov kernel and increases training time significantly. Therefore, we keep the SSM parameters fixed in our experiments.

Time-invariant Dynamics Both the UnHiPPO dynamics in Eq. (24) and the HiPPO dynamics in Eq. (25) are time-varying. However, the LSSL and UnLSSL layers fix the dynamics at a point in time and then apply them repeatedly. We can understand the effect of applying the dynamics in a time-invariant way visually from Fig. 5.

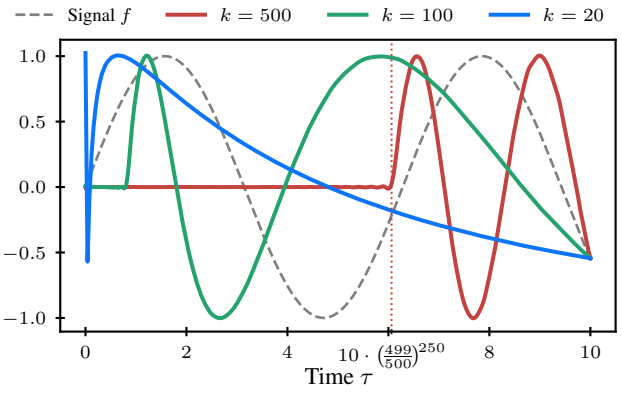

Figure 5. Reconstruction $\hat{f}_{\leq t_k}$ of signal f obtained through repeated application of $\bar{\boldsymbol{A}}_k$ and $\bar{\boldsymbol{B}}_k$ for fixed k. Data observed at 250 equispaced locations between 0 and 10. Each application of $\bar{\boldsymbol{A}}_k$ compresses the reconstruction $\hat{f}_{\leq t_k}$ by $^{k-1}/_{k}$.

Semantically, the dynamics $\bar{\boldsymbol{A}}_{k+1}$ of either HiPPO or UnHiPPO take the reconstruction $\hat{f}_{\leq t_k}$ of the signal from 0 to t_k represented by $\boldsymbol{c}_k$ and extend it by one step. Then, $\bar{\boldsymbol{B}}_{k+1}$ updates the extended reconstruction, so that it fits the data at t_{k+1}. To understand the effect of repeated application of the same $\bar{\boldsymbol{A}}_{k+1}$, it is instructive to imagine what happens if we map the signal back to the domain $[-1, 1]$ of the Legendre polynomials with ϕ_t from Section 3. In that view, $\bar{\boldsymbol{A}}_{k+1}$ squeezes the data that is encoded in $\hat{f}_{\leq t_k}$ as a function on

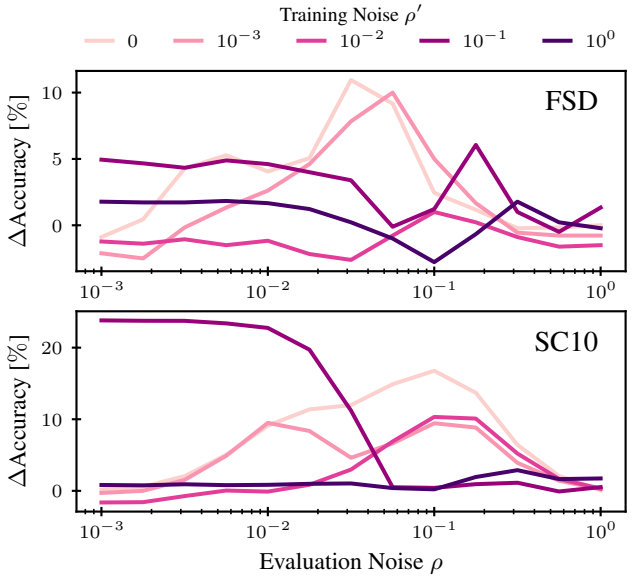

Figure 6. Accuracy difference between UnLSSL and LSSL in speech classification trained on a noise level ρ' and evaluated on a range of noise levels ρ.

$[-1, 1]$ into $[-1, 1-{}^{2}/_{k+1}]$ and $\bar{\boldsymbol{B}}_{k+1}$ fills the freed up interval $[1-{}^{2}/_{k+1}, 1]$ with the new data. So, repeated application of $\bar{A}_{k+1}$ squeezes the information encoded in $\hat{f}_{\leq t_k}$ by $^{k}/_{k+1}$ each time. Fig. 5 shows this as the 250-fold application of $\bar{\boldsymbol{A}}_{500}$ has compressed the initial value of $\hat{f}_{\leq t_k} = 0$ exactly by a factor of $({}^{499}/_{500})^{250}$. The fact that the $\hat{f}_{\leq t_k}$ for large k contain the whole signal while for small k they contain mostly the recent past, illuminates why initializing $\boldsymbol{A}$ and $\boldsymbol{B}$ with $\bar{\boldsymbol{A}}_k$ and $\bar{\boldsymbol{B}}_k$ for various k in LSSL and UnLSSL lets the model take multiple timescales into account as Gu et al. (2021) report.

6. Experiments

We experiment with a multi-layer LSSL and UnLSSL architecture with linear encoder and decoder as described by Gu et al. (2021). Section C gives further details on the hyperparameters.

We evaluate UnLSSL on two sequence classification datasets, the Free Spoken Digits dataset (FSD) (Jackson et al., 2018) and a 10-class subset of the Speech Commands dataset (SC10) (Warden, 2018). FSD contains 3000 recordings of 6 speakers pronouncing each digit from 0 to 9 at a sample rate of 8000. We loop recordings shorter than one second and then cut them so that each sample is a univariate sequence of length 8000. SC10 is a multi-speaker dataset consisting of one second clips each being a recording of one spoken word. Each sample is a univariate sequence of 16 000 steps representing the raw waveform of the audio.

For our experiments, where we measure the effect of noise in

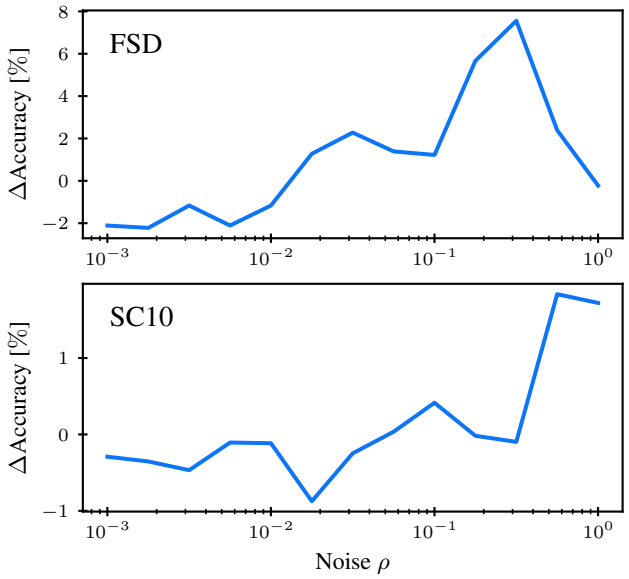

Figure 7. Accuracy difference between UnLSSL and LSSL in speech classification for models trained and evaluated on a range of noise levels ρ.

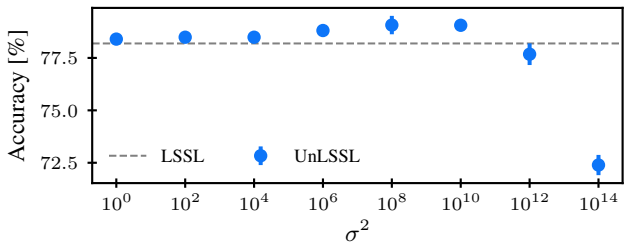

Figure 8. The σ^2 parameter adapts the noise filtering to the level of noise in the data on SC10 with $\rho = 0.1$. Error bars denote 1 standard deviation.

the data, we add independent Gaussian noise $\delta \sim \mathcal{N}(0, \rho^2)$ to the training and/or test data.

Noise Robustness For Fig. 6, we trained LSSL and UnLSSL on data with varying noise levels ρ' and evaluated the classification accuracy on a test set across a spectrum of noise levels ρ. The results show that the UnHiPPO initialization improves the model's robustness against other noise levels than it was trained on, both lower and higher.

Training Noise In the real world, training and evaluation data often have similar levels of noise. To investigate this case, we trained LSSL and UnLSSL on a range of noise levels ρ and then evaluated their accuracy on the test at the same noise level. Fig. 7 shows that the UnHiPPO initialization hinders the model's performance slightly at low noise levels while improving it as the noise level in the data increases.

Effect of σ^2 σ^2 in the Kalman filter in Eq. (23) influences how much noise the UnHiPPO dynamics filter out. This lets the user adapt UnHiPPO to the expected level of noise in the data. See Fig. 8 for an evaluation where we have trained UnLSSL on SC10 with Gaussian noise of $\rho = 10^{-1}$ and varied the σ^2 parameter. The results show that UnLSSL achieves a better fit on noisy data than LSSL.

Discretization Methods While LSSL relies on a trapezoidal-like discretization for the HiPPO matrix, we use the closed-form solution in terms of the matrix exponential for the UnHiPPO matrix in Eq. (21). Table 1 shows that the closed-form solution is only 50% slower than the trapezoidal approximation, which is negligible since this only needs to be computed once to initialize the SSM. An UnLSSL model initialized with the resulting UnHiPPO matrices reaches comparable performance for the closed-form solution and a stable backwards Euler approximation. Both the trapezoidal and forward Euler approximations produce an SSM initialization that diverges on the long sequences of SC10. Section B has a visualization of the effect of the discretization methods on the HiPPO and UnHiPPO matrices.

Table 1. Time to compute discretized $\bar{\boldsymbol{A}}_{\mathrm{R,k}}$ for $N = 256$ and resulting UnLSSL accuracy on SC10 with $\rho = 10^{-1}$ noise.

Method	Time	Accuracy
closed-form	2.8 ms	79.3%
trapezoidal	1.8 ms	-
forward	0.1 ms	-
backward	1.6 ms	79.8%

7. Related Work

Yu et al. (2023) introduce a diagonalization of the HiPPO matrix through an approach called "perturb-then-diagonalize". Their parametrization improves runtime performance over a non-diagonal one and is more robust against noise than the diagonalizations proposed by Smith et al. (2023); Gu et al. (2022b). Yu et al. (2025) propose an alternative parametrization for SSMs detached from HiPPO and instead based on Hankel operator theory. Their HOPE scheme improves the training stability of SSMs and makes them robust against noise appended to the end of a sequence. However, they do not evaluate noise on the data itself. Agarwal et al. (2023) derive an SSM from the spectral filtering algorithm (Hazan et al., 2017) which is based on an LDS with noisy outputs similar to our setup in Section 4. Zhou et al. (2024) describe an SSM that parametrizes a Kalman filter with a learnable dynamics matrix. They accelerate their model by rewriting the recurrence as a convolution with a Krylov kernel similar to (Gu et al., 2021). However, they do not connect it to HiPPO theory or SSM initialization in general.

Gu et al. (2022c) present additional background on the mechanism behind HiPPO. Gupta et al. (2022) explore a simplified alternative to HiPPO with a diagonal transition matrix. Liu & Li (2025) analyze the role of initializations in state space models discretized with the zero-order hold approximation.

8. Limitations

Gu et al. (2022a) have proposed an alternative parametrization of the HiPPO matrix $\boldsymbol{A}_{\mathrm{H}}$ in terms of a diagonal and a low-rank matrix, which they call a *structured* SSM. With this parametrization, the matrix multiplication in the SSM recurrence can be computed in $O(N)$ instead of $O(N^2)$. A similarly structured parametrization of either the regularized HiPPO matrix $\boldsymbol{A}_{\mathrm{R}}$ in Eq. (18) or the uncertainty-aware HiPPO matrix $\bar{\boldsymbol{A}}_{\mathrm{U},}$ in Eq. (24) eluded us, unfortunately, because of the pseudo-inverse and Kalman filter equations, respectively.

9. Conclusion

We analyzed HiPPO as a linear stochastic control problem and understood why it is not robust against noise. By rewriting the control problem as a system controlled by a latent state with noisy observations, we were able to derive the UnHiPPO dynamics that perform posterior inference in that system and thereby filter out noise. The resulting UnHiPPO initialization for SSMs offers a configurable amount of noise filtering without any modifications to the model structure or runtime costs. We have furthermore shown in our experiments how such built-in noise filtering can improve the robustness of SSMs against noise in the real world.

Acknowledgments

ML was funded by the Bavarian State Ministry for Science and the Arts within the framework of the Geothermal Alliance Bavaria project. We want to thank the Munich Center for Machine Learning for providing compute resources. Furthermore, we are thankful to Marcel Kollovieh for his insightful feedback and his support with Fig. 2.

Authors' Contributions

ML conceived the project idea, developed the theory, implemented the UnHiPPO initialization and wrote the manuscript. AS implemented the majority of the pipeline, designed and executed the empirical evaluation (i.e. selected the datasets and implemented model training and data loading to generate the quantitative results for visualization and analysis), identified a technical correction in the mathematical formulation, and corrected (Un)LSSL implementations. SG contributed to the design of the method and provided scientific guidance. All authors reviewed and discussed results, both theoretical and empirical, throughout the project.

Software

For our results, we rely on excellent software packages, notably `numpy` (Harris et al., 2020), `scipy` (Virtanen et al., 2020), `pytorch` (Paszke et al., 2019), `einops` (Rogozhnikov, 2022), `matplotlib` (Hunter, 2007), `hydra` (Yadan, 2019) and `jupyter` (Granger & Pérez, 2021).

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none of which we feel must be specifically highlighted here.

References

Agarwal, N., Suo, D., Chen, X., and Hazan, E. Spectral State Space Models, 2023.

Arfken, G. and Weber, H. J. *Mathematical Methods for Physicists*. Elsevier Academic Press, Amsterdam, Heidelberg, 6. ed., 5. [print., international ed.] edition, 2008. ISBN 978-0-12-059876-2.

Aström, K. J. *Introduction to Stochastic Control Theory*. Number v. 70 in Mathematics in Science and Engineering. Academic Press, New York, 1970. ISBN 978-0-12-065650-9.

Cho, K., van Merrienboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation, September 2014.

Goel, K., Gu, A., Donahue, C., and Re, C. It's Raw! Audio Generation with State-Space Models. In *International Conference on Machine Learning*, pp. 7616–7633. PMLR, June 2022.

Goldberg, D. What every computer scientist should know about floating-point arithmetic. *ACM Computing Surveys*, 23(1):5–48, March 1991. ISSN 0360-0300. doi: 10.1145/103162.103163.

Granger, B. E. and Pérez, F. Jupyter: Thinking and Storytelling With Code and Data. *Computing in Science & Engineering*, 23(2):7–14, March 2021. ISSN 1558-366X. doi: 10.1109/MCSE.2021.3059263.

Gu, A. and Dao, T. Mamba: Linear-Time Sequence Modeling with Selective State Spaces, December 2023.

Gu, A., Dao, T., Ermon, S., Rudra, A., and Re, C. HiPPO: Recurrent Memory with Optimal Polynomial Projections. In *Neural Information Processing Systems*, 2020.

Gu, A., Johnson, I., Goel, K., Saab, K., Dao, T., Rudra, A., and Ré, C. Combining Recurrent, Convolutional, and Continuous-time Models with Linear State-Space Layers. In *Neural Information Processing Systems*. arXiv, October 2021. doi: 10.48550/arXiv.2110.13985.

Gu, A., Goel, K., and Ré, C. Efficiently Modeling Long Sequences with Structured State Spaces. In *International Conference on Learning Representations*. arXiv, 2022a.

Gu, A., Gupta, A., Goel, K., and Ré, C. On the Parameterization and Initialization of Diagonal State Space Models, June 2022b.

Gu, A., Johnson, I., Timalsina, A., Rudra, A., and Re, C. How to Train your HIPPO: State Space Models with Generalized Orthogonal Basis Projections. In *International Conference on Learning Representations*, September 2022c.

Gupta, A., Gu, A., and Berant, J. Diagonal State Spaces are as Effective as Structured State Spaces. In *Advances in Neural Information Processing Systems*, October 2022.

Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., van Kerkwijk, M. H., Brett, M., Haldane, A., del Río, J. F., Wiebe, M., Peterson, P., Gérard-Marchant, P., Sheppard, K., Reddy, T., Weckesser, W., Abbasi, H., Gohlke, C., and Oliphant, T. E. Array programming with NumPy. *Nature*, 585(7825):357–362, September 2020. doi: 10.1038/s41586-020-2649-2.

Hazan, E., Singh, K., and Zhang, C. Learning Linear Dynamical Systems via Spectral Filtering. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R. (eds.), *Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

Hochreiter, S. and Schmidhuber, J. Long Short-Term Memory. *Neural Computation*, 9:1735–1780, 1997.

Hunter, J. D. Matplotlib: A 2D graphics environment. *Computing in Science & Engineering*, 9(3):90–95, 2007. doi: 10.1109/MCSE.2007.55.

Jackson, Z., Souza, C., Flaks, J., Pan, Y., Nicolas, H., and Thite, A. Free-spoken-digit-dataset: V1.0.8. Zenodo, August 2018.

Kalman, R. E. A New Approach to Linear Filtering and Prediction Problems. *Journal of Basic Engineering*, 82 (1):35–45, March 1960. ISSN 0021-9223. doi: 10.1115/1.3662552.

Liu, F. and Li, Q. Autocorrelation Matters: Understanding the Role of Initialization Schemes for State Space Models. In *International Conference on Learning Representations*, 2025.

Mehta, H., Gupta, A., Cutkosky, A., and Neyshabur, B. Long Range Language Modeling via Gated State Spaces. In *The Eleventh International Conference on Learning Representations*, September 2022.

Park, J., Kim, H.-S., Ko, K., Kim, M., and Kim, C. VideoMamba: Spatio-Temporal Selective State Space Model. In Leonardis, A., Ricci, E., Roth, S., Russakovsky, O., Sattler, T., and Varol, G. (eds.), *European Conference on Computer Vision*, pp. 1–18, Cham, 2025. Springer Nature Switzerland. ISBN 978-3-031-72698-9. doi: 10.1007/978-3-031-72698-9_1.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Köpf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Neural Information Processing Systems*, 2019.

Patro, B. N. and Agneeswaran, V. S. Mamba-360: Survey of State Space Models as Transformer Alternative for Long Sequence Modelling: Methods, Applications, and Challenges, April 2024a.

Patro, B. N. and Agneeswaran, V. S. SiMBA: Simplified Mamba-Based Architecture for Vision and Multivariate Time series, April 2024b.

Rogozhnikov, A. Einops: Clear and Reliable Tensor Manipulations with Einstein-like Notation. In *International Conference on Learning Representations*, 2022.

Särkkä, S. and Solin, A. *Applied Stochastic Differential Equations*. Cambridge University Press, April 2019. ISBN 978-1-108-18673-5.

Smith, J. T. H., Warrington, A., and Linderman, S. W. Simplified State Space Layers for Sequence Modeling. In *International Conference on Learning Representations*, 2023.

Vetterli, M., Kovačević, J., and Goyal, V. K. *Foundations of Signal Processing*. Cambridge University Press, Cambridge, 2014. ISBN 978-1-107-03860-8.

Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., van der Walt, S. J., Brett, M.,

Wilson, J., Millman, K. J., Mayorov, N., Nelson, A. R. J., Jones, E., Kern, R., Larson, E., Carey, C. J., Polat, İ., Feng, Y., Moore, E. W., VanderPlas, J., Laxalde, D., Perktold, J., Cimrman, R., Henriksen, I., Quintero, E. A., Harris, C. R., Archibald, A. M., Ribeiro, A. H., Pedregosa, F., van Mulbregt, P., and SciPy 1.0 Contributors. SciPy 1.0: Fundamental algorithms for scientific computing in python. *Nature Methods*, 17:261–272, 2020. doi: 10.1038/s41592-019-0686-2.

Voelker, A., Kajić, I., and Eliasmith, C. Legendre Memory Units: Continuous-Time Representation in Recurrent Neural Networks. In *Neural Information Processing Systems*, 2019.

Warden, P. Speech Commands: A Dataset for Limited-Vocabulary Speech Recognition, April 2018.

Yadan, O. Hydra - A framework for elegantly configuring complex applications. Github, 2019.

Yu, A., Nigmetov, A., Morozov, D., Mahoney, M. W., and Erichson, N. B. Robustifying State-space Models for Long Sequences via Approximate Diagonalization. In *International Conference on Learning Representations*. arXiv, October 2023. doi: 10.48550/arXiv.2310.01698.

Yu, A., Mahoney, M. W., and Erichson, N. B. HOPE for a Robust Parameterization of Long-memory State Space Models. In *International Conference on Learning Representations*. arXiv, 2025. doi: 10.48550/arXiv.2405.13975.

Zhou, Z., Guo, X., Jie, Y., and Xia, C.-M. Kalman-SSM: Modeling long-term time series with kalman filter structured state spaces. *IEEE Signal Processing Letters*, PP: 1–5, January 2024. doi: 10.1109/LSP.2024.3457862.

A. HiPPO Derivation

A complete derivation can, of course, also be found in the original work of (Gu et al., 2020). However, the original derivation includes several additional degrees of freedom to support multiple HiPPO variants. Since the variant based on Legendre polynomials is the only one relevant to our work and also the one most relevant to follow-up works (Gu et al., 2021; 2022a), we give a specialized derivation here that eschews unnecessary symbols.

A.1. Choice of Basis

Let's consider the case where we observe a continuous scalar signal $f(\tau) : \mathbb{R}^+ \to \mathbb{R}$ and at any point t in time we want to encode the history $f_{\leq t} \coloneqq f|_{[0,t]}$ of the signal that we have observed so far into an N dimensional vector $\boldsymbol{c}_t$. We consider $\boldsymbol{c}_t$ a good representation of $f_{\leq t}$, if we can turn it back into an approximate version $\hat{f}_{\leq t}$ of $f_{\leq t}$, we can rely on the polynomial approximation framework for signal processing (Vetterli et al., 2014). In this framework, we begin by choosing an inner product $\langle \cdot, \cdot \rangle$ between functions on $[0, t]$ to induce a distance via $d(f, g) = \sqrt{\|f - g\|}$ and $\|f - g\| = \langle f - g, f - g \rangle$ and thus define exactly what $\hat{f}_{\leq t} \approx f_{\leq t}$ should mean. Equipped with this inner product, we can then derive a set of N orthonormal polynomials $g_i, i = 0, \ldots, N-1$ on $[0, t]$ to form the basis of a finite-dimensional function space in which we can approximate the continuous signal $f_{\leq t}$. Finally, we can find $\boldsymbol{c}_t$ by projecting $f_{\leq t}$ onto g_i, i.e. $c_{t,i} = \langle f_{\leq t}, g_i \rangle$.

Our derivation of the Legendre-based HiPPO begins with an inner product between two functions f and g on $[0, t]$. We will choose

$$\langle f, g \rangle_t = \frac{1}{t} \int_0^t f(\tau)\, g(\tau)\, \mathrm{d}\tau, \tag{30}$$

which is just the L^2 inner product between functions except for a constant factor $1/t$. While the constant factor has no effect on orthogonality, it will be useful later when we update the representation $\boldsymbol{c}_t$ with new data. In terms of function approximation, this inner product induces the distance

$$\big(d(f, g)\big)^2 = \frac{1}{t} \int_0^t \big(f(\tau) - g(\tau)\big)^2\, \mathrm{d}\tau. \tag{31}$$

This means that the approximation $\hat{f}_{\leq t}$ we are looking for minimizes the squared difference to $f_{\leq t}$ across the segment from 0 to t with equal weight given to each τ.

Next, we need the polynomial basis. Since the Legendre polynomials P_i are an orthogonal basis on $[-1, 1]$ for the L^2 inner product, they are also orthogonal on $[0, t]$ with respect to our shifted and scaled inner product if we shift and scale them linearly from $[-1, 1]$ to $[0, t]$. For this shifting, we define $\phi_t(\tau) = 2\tau/t - 1$ that takes $\tau \in [0, t]$ to $x \in [-1, 1]$. This gives us $P_i(\phi_t(\tau))$ as an orthogonal basis on $[0, t]$. However, the projection of $f_{\leq t}$ onto the basis is simplified if the basis is orthonormal, so we need to compute the normalization constants of this shifted basis. With $\phi_t'(\tau) = 2/t$ and substitution integration, we get

$$\|P_i(\phi_t(\tau))\|_t^2 = \langle P_i(\phi_t(\tau)), P_i(\phi_t(\tau)) \rangle_t = \frac{1}{t} \int_0^t P_i(\phi_t(\tau))^2\, \mathrm{d}\tau = \frac{1}{2} \int_{-1}^1 P_i(x)^2\, \mathrm{d}x = \frac{1}{2i+1}. \tag{32}$$

The last step uses the normalization constant of the unscaled Legendre polynomials of $2/2i+1$. This gives us

$$g_{t,i}(\tau) = \sqrt{2i+1}\, P_i(\phi_t(\tau)), \quad i = 0, \ldots, D \tag{33}$$

as a basis of orthonormal polynomials on $[0, t]$ up to any degree D.

Computing the representation $\boldsymbol{c}_t \in \mathbb{R}^N$ thus means evaluating $c_{t,i} = \langle f_{\leq t}, g_{t,i} \rangle_t$ with Eq. (30) for $i = 0, \ldots, N-1$.

A.2. Linear Dynamics

How does the representation $\boldsymbol{c}_t$ of $f_{\leq t}$ change as we observe more of the signal? In the continuous case, the answer to this question is the ODE

$$\frac{\mathrm{d}c_{t,i}}{\mathrm{d}t} = \frac{\mathrm{d}}{\mathrm{d}t} \langle f_{\leq t}, g_{t,i} \rangle_t = \frac{\mathrm{d}}{\mathrm{d}t} \frac{1}{t} \int_0^t f(\tau)\, g_{t,i}(\tau)\, \mathrm{d}\tau = -\frac{1}{t^2} \int_0^t f(\tau)\, g_{t,i}(\tau)\, \mathrm{d}\tau + \frac{1}{t} \frac{\mathrm{d}}{\mathrm{d}t} \int_0^t f(\tau)\, g_{t,i}(\tau)\, \mathrm{d}\tau. \tag{34}$$

The first part is just $-1/t\, c_{t,i}$ but the derivative of the integral in the second part needs some more work. We can exchange differentiation and integration with Leibniz's rule and get

$$\frac{\mathrm{d}}{\mathrm{d}t}\int_0^t f(\tau)\, g_{t,i}(\tau)\,\mathrm{d}\tau = f(t)\, g_{t,i}(t) + \int_0^t f(\tau)\,\frac{\mathrm{d}g_{t,i}(\tau)}{\mathrm{d}t}\,\mathrm{d}\tau. \tag{35}$$

To progress, we need to use some properties of Legendre polynomials. For the first part, we use that $P_i(1) = 1$ for all i and therefore $g_{t,i}(t) = \sqrt{2i+1}$. For the second part, we use the identity (see (Gu et al., 2020, Eq. (8)) and (Arfken & Weber, 2008, Eqs. (12.23) and (12.24)))

$$(x+1)\,\frac{\mathrm{d}}{\mathrm{d}x}P_i(x) = iP_i(x) + \sum_{j=1}^{i}(2(i-j)+1)\,P_{i-j}(x) \tag{36}$$

to derive $\frac{\mathrm{d}}{\mathrm{d}t}g_{t,i}(\tau)$. Note that

$$\frac{\mathrm{d}}{\mathrm{d}t}\phi_t(\tau) = \frac{\mathrm{d}}{\mathrm{d}t}\left[\frac{2\tau}{t} - 1\right] = -\frac{2\tau}{t^2} = -\frac{1}{t}\left(\frac{2\tau}{t} - 1 + 1\right) = -\frac{1}{t}\big(\phi_t(\tau) + 1\big). \tag{37}$$

With Eqs. (36) and (37) and recognizing $\phi_t(\tau)$ as x, we get

$$\begin{aligned}
\frac{\mathrm{d}}{\mathrm{d}t}g_{t,i}(\tau) &= \sqrt{2i+1}\,\frac{\mathrm{d}}{\mathrm{d}t}P_i(\phi_t(\tau)) \\
&= \sqrt{2i+1}\,\frac{\mathrm{d}\phi_t(\tau)}{\mathrm{d}t}\,\frac{\partial}{\partial t}P_i(\phi_t(\tau)) \\
&= -\frac{1}{t}\sqrt{2i+1}\,\big(\phi_t(\tau)+1\big)\,\frac{\partial}{\partial t}P_i(\phi_t(\tau)) \\
&= -\frac{1}{t}\sqrt{2i+1}\left[iP_i(\phi_t(\tau)) + \sum_{j=1}^{i}(2(i-j)+1)\,P_{i-j}(\phi_t(\tau))\right] \\
&= -\frac{1}{t}\left[ig_{t,i}(\tau) + \sqrt{2i+1}\sum_{j=1}^{i}\sqrt{2(i-j)+1}\,g_{t,i-j}(\tau)\right].
\end{aligned} \tag{38}$$

When we plug this into the second part of Eq. (35), we see that

$$\begin{aligned}
\int_0^t f(\tau)\,\frac{\mathrm{d}g_{t,i}(\tau)}{\mathrm{d}t}\,\mathrm{d}\tau &= -\frac{1}{t}\int_0^t f(\tau)\left[ig_{t,i}(\tau) + \sqrt{2i+1}\sum_{j=1}^{i}\sqrt{2(i-j)+1}\,g_{t,i-j}(\tau)\right]\mathrm{d}\tau \\
&= -\left[i\,\underbrace{\langle f, g_{t,i}\rangle_t}_{c_{t,i}} + \sqrt{2i+1}\sum_{j=1}^{i}\sqrt{2(i-j)+1}\,\underbrace{\langle f, g_{t,i-j}\rangle_t}_{c_{t,i-j}}\right]
\end{aligned} \tag{39}$$

If we plug Eq. (39) into Eq. (35) and that in turn into Eq. (34), we see that

$$\begin{aligned}
\frac{\mathrm{d}c_{t,i}}{\mathrm{d}t} &= -\frac{1}{t^2}\int_0^t f(\tau)\, g_{t,i}(\tau)\,\mathrm{d}\tau + \frac{1}{t}\frac{\mathrm{d}}{\mathrm{d}t}\int_0^t f(\tau)\, g_{t,i}(\tau)\,\mathrm{d}\tau \\
&= -\frac{1}{t}c_{t,i} + \frac{1}{t}\left[\sqrt{2i+1}f(t) - \left[i\,c_{t,i} + \sqrt{2i+1}\sum_{j=1}^{i}\sqrt{2(i-j)+1}\,c_{t,i-j}\right]\right].
\end{aligned} \tag{40}$$

Now we can separate terms involving $\boldsymbol{c}$ and f and write the N instances of Eq. (40) for the N components of $\boldsymbol{c}$ as a single vector differential equation. This gives us the full continuous coefficient dynamics from (Gu et al., 2020)

$$\frac{\mathrm{d}\boldsymbol{c}_t}{\mathrm{d}t} = -\frac{1}{t}\boldsymbol{A}_{\mathrm{H}}\boldsymbol{c}_t + \frac{1}{t}\boldsymbol{B}_{\mathrm{H}}f(t) \tag{41}$$

with the HiPPO matrix and vector

$$A_{\mathrm{H,ij}} = \begin{cases} \sqrt{2i+1}\sqrt{2j+1} & \text{if } j < i \\ i+1 & \text{if } j = i \\ 0 & \text{if } j > i \end{cases} \qquad \text{and} \qquad B_{\mathrm{H,i}} = \sqrt{2i+1}, \tag{42}$$

respectively.

A.3. Discretization

Usually, we observe the signal f at discrete time points t_k instead of truly continuously. Therefore, we cannot evaluate the exact, continuous dynamics in Eq. (41) and have to discretize them instead, so that we only ever need to evaluate f at the discrete observation points t_k.

With a continuous signal, we would compute $\boldsymbol{c}_{t_{k+1}}$ from c_{t_k} by integrating Eq. (41) from t_k to t_{k+1}, i.e.

$$\boldsymbol{c}_{t_{k+1}} = \boldsymbol{c}_{t_k} + \int_{t_k}^{t_{k+1}} -\frac{1}{t}\boldsymbol{A}_{\mathrm{H}}\boldsymbol{c}_t + \frac{1}{t}\boldsymbol{B}_{\mathrm{H}} f(t)\,\mathrm{d}t. \tag{43}$$

For discrete observations $f(t_1), f(t_2), \ldots$, we need to approximate the integral on the right-hand side in a way that only requires $f(t_k)$ and $f(t_{k+1})$.

(Gu et al., 2020) describe three approximations for the integral, *forward Euler*, *backward Euler* and the *trapezoidal rule* of which they use the latter for their experiments. Forward Euler approximates the value of the integrand with its value at the lower integration bound, i.e. at t_k, and gives

$$\boldsymbol{c}_{t_{k+1}} \approx \boldsymbol{c}_{t_k} + \underbrace{(t_{k+1} - t_k)}_{=:\Delta t}\left(-\frac{1}{t_k}\boldsymbol{A}_{\mathrm{H}}\boldsymbol{c}_{t_k} + \frac{1}{t_k}\boldsymbol{B}_{\mathrm{H}} f(t_k)\right) = \left(\boldsymbol{I} - \frac{\Delta t}{t_k}\boldsymbol{A}_{\mathrm{H}}\right)\boldsymbol{c}_{t_k} + \frac{\Delta t}{t_k}\boldsymbol{B}_{\mathrm{H}} f(t_k). \tag{44}$$

Backward Euler is the other extreme that approximates the integrand with its value at t_{k+1}, i.e.

$$\boldsymbol{c}_{t_{k+1}} \approx \boldsymbol{c}_{t_k} + \Delta t\left(-\frac{1}{t_{k+1}}\boldsymbol{A}_{\mathrm{H}}\boldsymbol{c}_{t_{k+1}} + \frac{1}{t_{k+1}}\boldsymbol{B}_{\mathrm{H}} f(t_{k+1})\right). \tag{45}$$

Solving for $\boldsymbol{c}_{t_{k+1}}$ gives us

$$\boldsymbol{c}_{t_{k+1}} \approx \left(\boldsymbol{I} + \frac{\Delta t}{t_{k+1}}\boldsymbol{A}_{\mathrm{H}}\right)^{-1}\left[\boldsymbol{c}_{t_k} + \frac{\Delta t}{t_{k+1}}\boldsymbol{B}_{\mathrm{H}} f(t_{k+1})\right]. \tag{46}$$

For the trapezoidal rule, we approximate the integrand as the average of its values at t_k and t_{k+1}, i.e.

$$\boldsymbol{c}_{t_{k+1}} \approx \boldsymbol{c}_{t_k} + \Delta t\,\frac{1}{2}\left[\left(-\frac{1}{t_k}\boldsymbol{A}_{\mathrm{H}}\boldsymbol{c}_{t_k} + \frac{1}{t_k}\boldsymbol{B}_{\mathrm{H}} f(t_k)\right) + \left(-\frac{1}{t_{k+1}}\boldsymbol{A}_{\mathrm{H}}\boldsymbol{c}_{t_{k+1}} + \frac{1}{t_{k+1}}\boldsymbol{B}_{\mathrm{H}} f(t_{k+1})\right)\right]. \tag{47}$$

After simplifying and solving for $\boldsymbol{c}_{t_{k+1}}$, this becomes

$$\boldsymbol{c}_{t_{k+1}} \approx \left(\boldsymbol{I} + \frac{\Delta t}{2t_{k+1}}\boldsymbol{A}_{\mathrm{H}}\right)^{-1}\left[\left(\boldsymbol{I} - \frac{\Delta t}{2t_k}\boldsymbol{A}_{\mathrm{H}}\right)\boldsymbol{c}_{t_k} + \frac{\Delta t}{2}\boldsymbol{B}_{\mathrm{H}}\left(\frac{1}{t_k} f(t_k) + \frac{1}{t_{k+1}} f(t_{k+1})\right)\right]. \tag{48}$$

Note that (Gu et al., 2020) approximate Eq. (48) as

$$\boldsymbol{c}_{t_{k+1}} \approx \left(\boldsymbol{I} + \frac{\Delta t}{2t_{k+1}}\boldsymbol{A}_{\mathrm{H}}\right)^{-1}\left[\left(\boldsymbol{I} - \frac{\Delta t}{2t_{k+1}}\boldsymbol{A}_{\mathrm{H}}\right)\boldsymbol{c}_{t_k} + \frac{\Delta t}{t_{k+1}}\boldsymbol{B}_{\mathrm{H}} f(t_{k+1})\right], \tag{49}$$

i.e. they approximate $1/t_k f(t_k) + 1/t_{k+1} f(t_{k+1})$ as $2/t_{k+1} f(t_{k+1})$. This, in effect, pretends that the dynamics in Eq. (41) would be constant in time and is required by LSSL for the numerical stability of the accelerated evaluation via convolution with a Krylov kernel. In this work, we discretize the LSSL baseline with Eq. (49), too. Note that for our model a closed-form solution in terms of the matrix exponential exists, which we use instead of a numerical approximation.

Each of Eqs. (44), (46), (48) and (49) can be put into the form

$$\boldsymbol{c}_{t_{k+1}} = \bar{\boldsymbol{A}}_{\mathrm{H},t_k}\boldsymbol{c}_{t_k} + \bar{\boldsymbol{B}}_{\mathrm{H},t_k} f(t_k) + \bar{\boldsymbol{B}}_{\mathrm{H},t_{k+1}} f(t_{k+1}) \tag{50}$$

by collecting terms appropriately.

B. Discretizations Visualized

Recurrent roll-out of an SSM or the construction of the Krylov kernel equate to repeated matrix multiplication with the HiPPO or UnHiPPO matrix or taking powers thereof. Therefore, successful training and evaluation on long sequences requires the discretization to be chosen so that matrix powers do not diverge. Fig. 9 shows the effect of repeated applications of HiPPO and UnHiPPO matrices discretized with the closed-form solution and the approximations considered by (Gu et al., 2021). The HiPPO matrix is stable under the backward discretizations and has a negligible degree of divergence with the trapezoidal rule. The UnHiPPO matrix cannot be stably discretized with the trapezoidal rule, but its closed-form discretization is stable and retains the information encoded in the shape of the reconstruction $\hat{f}$ almost perfectly.

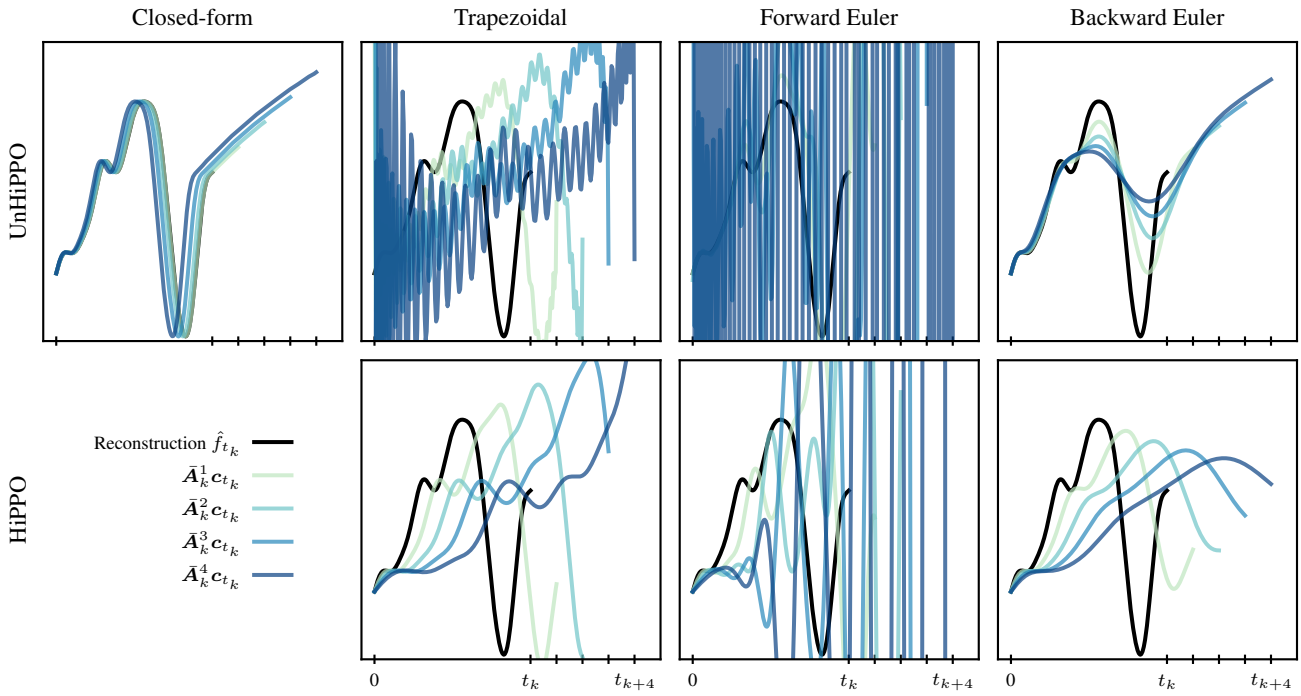

Figure 9. Visualization of repeated application of HiPPO and UnHiPPO matrices with various discretization methods through the polynomial representation $\hat{f}$ of $\boldsymbol{c}$. Magnitude of the plotted curves correlates with $\|\bar{\boldsymbol{A}}_k^i \boldsymbol{c}_{t_k}\|$.

C. Experiment Details

We use the parameters listed in Table 2 for all models. For the range of t_k in the initialization of LSSL, we set $t_{\min} = 10$ and $t_{\max} = 1000$ to cover a range of time scales.

Table 2. Hyperparameters of the LSSL architecture used for the SC10 experiments.

Parameter	Value
Layers	4
N	128
Linear Embedding Size	128
Latent Channels	4
Dropout	0.1
UnHiPPO σ^2	10^{10}
Training Steps	100000
Batch Size	16