KalMamba: Towards Efficient Probabilistic State Space Models for RL under Uncertainty

Philipp Becker[*] **Niklas Freymuth** **Gerhard Neumann**
Karlsruhe Institute of Technology

Abstract

Probabilistic State Space Models (SSMs) are essential for Reinforcement Learning (RL) from high-dimensional, partial information as they provide concise representations for control. Yet, they lack the computational efficiency of their recent deterministic counterparts such as *S4* or *Mamba*. We propose *KalMamba*, an efficient architecture to learn representations for RL that combines the strengths of probabilistic SSMs with the scalability of deterministic SSMs. *KalMamba* leverages *Mamba* to learn the dynamics parameters of a linear Gaussian SSM in a latent space. Inference in this latent space amounts to standard Kalman filtering and smoothing. We realize these operations using parallel associative scanning, similar to *Mamba*, to obtain a principled, highly efficient, and scalable probabilistic SSM. Our experiments show that *KalMamba* competes with state-of-the-art SSM approaches in RL while significantly improving computational efficiency, especially on longer interaction sequences.

1 Introduction

Deep probabilistic State Space Models (SSMs) are versatile tools widely used in Reinforcement Learning (RL) for environments with high-dimensional, partial, or noisy observations [22, 34, 38, 5, 25, 41]. They model states and observations as random variables and relate them through a set of conditional distributions, allowing them to capture uncertainties and learn concise probabilistic representations for downstream RL applications. Beyond RL, recent deterministic SSMs [16, 48, 15] offer a powerful new paradigm for general sequence modeling and rival state-of-the-art transformers while improving computational complexity [15]. These models assume states and observations are vectors related by deterministic, linear, and associative functions, which allow efficient time-parallel computations. Such deterministic models are often insufficient for RL with complex observations, where uncertainty awareness and probabilistic modeling are crucial [10, 34, 23]. In contrast, due to their nonlinear parameterizations and inference approaches, most existing probabilistic SSMs for RL and beyond do not feature the favorable scaling behavior of recent deterministic SSMs.

Many real-world applications require both uncertainty awareness and the capability of handling long sequences. Examples include multi-modal robotics tasks with high-frequency control, long sequence non-stationary tasks, or complex information-gathering tasks. Consider a robot tasked with packing objects of unknown properties into a basket. By interacting with each item to infer and memorize properties such as mass and deformability, the robot refines its understanding of the scene, enabling it to strategically arrange the objects in the basket. Current deterministic SSMs lack uncertainty awareness to solve such tasks, while their probabilistic counterparts do not scale to the required sequence lengths. Thus, the question of how to develop a principled method that combines the benefits of both paradigms to obtain robust and efficient probabilistic state space models for long-sequence RL under uncertainty arises.

[*]Correspondence to `philipp.becker@kit.edu`.

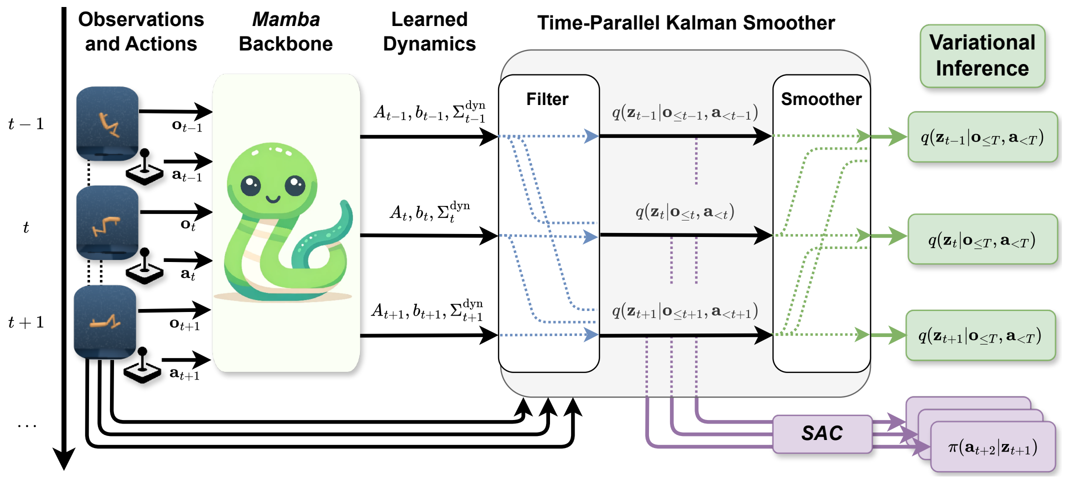

Observations and Actions
Mamba Backbone
Learned Dynamics
Time-Parallel Kalman Smoother
Variational Inference
Filter
Smoother
SAC
$t-1$
t
$t+1$
$\dots$
$\mathbf{o}_{t-1}$
$\mathbf{a}_{t-1}$
$\mathbf{o}_t$
$\mathbf{a}_t$
$\mathbf{o}_{t+1}$
$\mathbf{a}_{t+1}$
$A_{t-1}, b_{t-1}, \Sigma^{\text{dyn}}_{t-1}$
$A_t, b_t, \Sigma^{\text{dyn}}_t$
$A_{t+1}, b_{t+1}, \Sigma^{\text{dyn}}_{t+1}$
$q(\mathbf{z}_{t-1}|\mathbf{o}_{\leq t-1}, \mathbf{a}_{<t-1})$
$q(\mathbf{z}_t|\mathbf{o}_{\leq t}, \mathbf{a}_{<t})$
$q(\mathbf{z}_{t+1}|\mathbf{o}_{\leq t+1}, \mathbf{a}_{<t+1})$
$q(\mathbf{z}_{t-1}|\mathbf{o}_{\leq T}, \mathbf{a}_{<T})$
$q(\mathbf{z}_t|\mathbf{o}_{\leq T}, \mathbf{a}_{<T})$
$q(\mathbf{z}_{t+1}|\mathbf{o}_{\leq T}, \mathbf{a}_{<T})$
$\pi(\mathbf{a}_{t+2}|\mathbf{z}_{t+1})$

Figure 1: Overview of *KalMamba*. The observation-action sequences are first fed through a dynamics backbone built on *Mamba* [15] to learn a linear dynamics model for each step. *KalMamba* then uses time-parallel Kalman filtering [42] to infer filtered beliefs $q(\mathbf{z}_t|\mathbf{o}_{\leq t}, \mathbf{a}_{\leq t-1})$ which can be used for control with a *Soft Actor Critic (SAC)* [21]. For model training, *KalMamba* employs an additional time-parallel Kalman smoothing step to obtain smoothed beliefs $q(\mathbf{z}_t|\mathbf{o}_{\leq T}, \mathbf{a}_{\leq T})$. These beliefs allow training a model that excels in modeling uncertainties due to a tight variational lower bound [5]. Crucially, the smoothing step does not introduce trainable model parameters, enabling the direct use of the filtered beliefs for downstream RL policy training and execution.

We propose an efficient architecture for RL that equips probabilistic SSMs with the efficiency of recent deterministic SSMs. Our approach, *KalMamba*, uses (extended) Kalman filtering and smoothing [28, 40, 27] to infer belief states over a linear Gaussian SSM in a latent space that uses a dynamics model based on *Mamba* [15]. In this approach, Mamba acts as a highly effective general-purpose sequence-to-sequence model to learn the parameters of a dynamics model. The Kalman Smoother uses this model to compute probabilistic beliefs over system states. Figure 1 provides a schematic overview. *Mamba* is efficient for long sequences as it uses parallel associative scans, which allow parallelizing associative operators on highly parallel hardware accelerators such as GPUs [44]. Similarly, we formulate both Kalman filtering and smoothing as associative operations [42] and build efficient parallel scans for filtering and smoothing in PyTorch [39]. With both *Mamba* and the Kalman Smoother being parallelizable, *KalMamba* achieves time-parallel computation of belief states required for model learning and control. Thus, unlike previous approaches for efficient SSM-based RL [41], which rely on simplified inference assumptions, *KalMamba* enables end-to-end model training under high levels of uncertainty using a smoothing inference and tight variational lower bound [5]. While using smoothed beliefs for model learning, our architecture ensures a tight coupling between filtered and smoothed belief states. This inductive bias ensures the filtered beliefs are meaningful, allowing their use for policy learning and execution where future observations are unavailable.

We evaluate *KalMamba* on several tasks from the DeepMind Control (DMC) Suite [50], training an off-the-shelf *Soft Actor-Critic* [21] on beliefs inferred from both images and states. As baselines, we compare to *Recurrent State Space Models* [23] and the *Variational Recurrent Kalman Network* [5]. Our preliminary experiments show that *KalMamba* is competitive to these state-of-the-art SSMs while being significantly faster to train and scaling gracefully to long sequences due to its ability to be efficiently parallelized. These results indicate *KalMamba*'s potential for applications that require forming accurate belief states over long sequences under uncertainty.

To summarize our contributions, we (i) propose *KalMamba*, a novel probabilistic SSM for RL that combines Kalman filtering, smoothing, and a Mamba backbone to offer efficient probabilistic inference, (ii) motivate and compare *KalMamba* to existing probabilistic SSMs for RL, and (iii) validate our approach on state- and image-based control tasks, closely matching the performance of state-of-the-art probabilistic SSMs while being time-parallelizable.

2 Related Work

Deterministic State Space Models in Deep Learning. Structured deterministic State Space approaches [16, 48, 15] recently emerged as an alternative to the predominant Transformer [52] architecture for general sequence modeling [15]. Their main benefit is combining compute and memory requirements that scale linearly in sequence length with efficient and parallelizable implementations. While earlier approaches, such as the *Structured State Space Sequence Model (S4)* [16] and others [18, 26] used a convolutional formulation for efficiency, more recent approaches [48, 15] use associative scans. Such associative scans allow for parallel computations over sequences if all involved operators are associative, which yields a logarithmic runtime, given enough parallel cores. However, all these models are deterministic, i.e., they do not model uncertainties or allow sampling without further modifications. As a remedy, *Latent S4 (LS4)* [56] extends *S4* for probabilistic generative sequence modeling and forecasting. However, in LS4, the latent states are not Markovian and are thus hard to use for control. *KalMamba* exploits the fact that filtering and smoothing in linear Gaussian state space models can also be formulated as a set of associative operations, which makes it amenable to parallel scans [42]. To our knowledge, it is the first deep-learning model to do so. Further, it relies on *Mamba* [15], a state-of-the-art deterministic state space model, to precompute the dynamics models required for filtering and smoothing.

Probabilistic State Space Models for Reinforcement Learning. Probabilistic state space models are commonly and successfully used for reinforcement learning from high dimensional or multimodal observations [38, 54, 25, 4], under partial observability [5], and for memory tasks [41]. Arguably, the most prominent approach is the *Recurrent State Space Model (RSSM)* [23]. After their original introduction as the basis of a standard planner, they have been improved with more involved parametric policy learning approaches [22] and categorical latent variables for categorical domains [24]. During inference, the *RSSMs* conditions the latent state on past observations and actions, resulting in a filtering inference scheme. Here, the key architectural feature of *RSSMs* is splitting the latent state into stochastic and deterministic parts. The deterministic part is then propagated through time using a standard recurrent architecture. In its original formulation, the *RSSM* uses a Gated Recurrent Unit (GRU) [9]. One line of research focuses on replacing this deterministic path with more efficient architectures with the *TransDreamer* [8] approach using a transformer [52] and *Recall to Image* [41] using S4 [16]. However, to fully exploit the efficiency of these backbone architectures, both need to simplify the inference assumptions and can only consider the current observation, which makes them highly susceptible to noise or missing observations. Opposed to that, the *Variational Recurrent Kalman Network (VRKN)* [5] proposes using a smoothing inference scheme that conditions both past and future actions. This scheme allows the *VRKN* to work with a fully stochastic latent state and lets it excel in tasks where modeling uncertainty is crucial. The *VRKN* uses a locally linear Gaussian State Space Model in a latent space, performing closed-form Kalman Filtering and smoothing. *KalMamba* holistically combines smoothing inference in a fully probabilistic SSM with an efficient temporally parallelized implementation, resulting in an approach that is robust to noise and efficient.

Probabilistic State Space Models in Deep Learning. Probabilistic state space models are versatile and commonly used tools in machine learning. Besides classical approaches using linear models [47] and works using Gaussian Processes [12, 11], most recent methods build on Neural Networks (NNs) to parameterize generative and inference models using the SSM assumptions [2, 53, 17, 29, 13, 33, 3, 55, 43, 37, 6, 7, 35, 45, 32, 46]. Out of these approaches, those that embed linear-Gaussian SSMs into latent spaces [53, 20, 13, 3, 7, 6, 45, 32, 46] are of particular relevance to *KalMamba*. Doing so allows for closed-form inference using (extended) Kalman Filtering and Smoothing. However, with the notable exception of the *VRKN*, these models usually cannot be used to control or even model systems of similar complexity to those controlled with *RSSM*-based approaches. Furthermore, some of them [29, 7] do not allow smoothing, while others [13, 32] model observations in the latent space as additional random variables which complicates inference and training and prevents principled usage of the observation uncertainty for filtering. Another class of approaches [20, 6, 45, 46] trains using regression and are thus not generative. Notably, none of these approaches uses a temporally parallelized formulation of the filtering and smoothing operations.*KalMamba* takes inspiration from many of these approaches and partly follows the *VRKN*'s design to enable reinforcement learning for complex systems. However, it combines those ideas with the efficiency of recent deterministic SSMs using an architecture that enables time-parallel computations.

3 State Space Models for Reinforcement Learning

In Reinforcement Learning (RL) under uncertainty and partial observability, State Space Models (SSMs) generally assume sequences of observations $\mathbf{o}_{\leq T} = \{\mathbf{o}_t\}_{t=0\cdots T}$ which are generated by a sequence of latent state variables $\mathbf{z}_{\leq T} = \{\mathbf{z}_t\}_{t=0\cdots T}$, given a sequence of actions $\mathbf{a}_{\leq T} = \{\mathbf{a}_t\}_{t=0\cdots T}$. The corresponding generative model factorizes according to the hidden Markov assumptions [36], i.e., each observation $\mathbf{o}_t$ only depends on the current latent state $\mathbf{z}_t$ through an observation model $p(\mathbf{o}_t|\mathbf{z}_t)$, and each latent state $\mathbf{z}_t$ only depends on the previous state $\mathbf{z}_{t-1}$ and the action $\mathbf{a}_{t-1}$ through a dynamics model $p(\mathbf{z}_t|\mathbf{z}_{t-1}, \mathbf{a}_{t-1})$.

In order to learn the state space model from data and use it for downstream RL, we need to infer latent belief states given observations and actions. Depending on the information provided for inference, we differentiate between the filtered belief $\mathbf{q}(\mathbf{z}_t|\mathbf{o}_{\leq t}, \mathbf{a}_{\leq t-1})$ and the smoothed belief $\mathbf{q}(\mathbf{z}_t|\mathbf{o}_{\leq T}, \mathbf{a}_{\leq T})$. The filtered belief conditions only on past information, while the smoothed belief also depends on future information. Computing these beliefs is intractable for models of reasonable complexity. Thus, we resort to an autoencoding variational Bayes approach that allows joint training of the generative and an approximate inference model using a lower bound objective [31].

The *Recurrent State Space Model (RSSM)* [23] assumes a nonlinear dynamics model, splitting the state $\mathbf{z}_t$ into a stochastic $\mathbf{s}_t$ and a deterministic part $\mathbf{h}_t$ which evolve according to $\mathbf{h}_t = f(\mathbf{h}_{t-1}, \mathbf{a}_{t-1}, \mathbf{s}_{t-1})$ and $\mathbf{s}_t \sim p(\mathbf{s}_t|\mathbf{h}_t)$. Here f is implemented using a *Gated Recurrent Unit (GRU)* [9]. This results in a nonlinear, autoregressive process that cannot be parallelized over time. Further, *RSSMs* assume a filtering inference model $q(\mathbf{s}_t|\mathbf{h}_t, \mathbf{o}_t)$, where $\mathbf{h}_t$ accumulates all information from the past. The *RSSM*'s inference scheme struggles with correctly estimating uncertainties as the resulting lower bound is not tight [5]. In tasks where such uncertainties are relevant, this lack of principled uncertainty estimation causes poor performance for downstream applications.

Table 1: Comparing the inference models and capabilities for smoothing (Smooth) and time-parallel (Parallel) execution of recent SSMs for RL.

Method	Inference Model	Smooth	Parallel
RSSM [23]	$q(\mathbf{z}_t\|\mathbf{h}_t, \mathbf{o}_t)$	×	×
R2I [41]	$q(\mathbf{z}_t\|\mathbf{o}_t)$	×	✓
VRKN [5]	$q(\mathbf{z}_t\|\mathbf{o}_{\leq T}, \mathbf{a}_{\leq T})$	✓	×
KalMamba	$q(\mathbf{z}_t\|\mathbf{o}_{\leq T}, \mathbf{a}_{\leq T})$	✓	✓

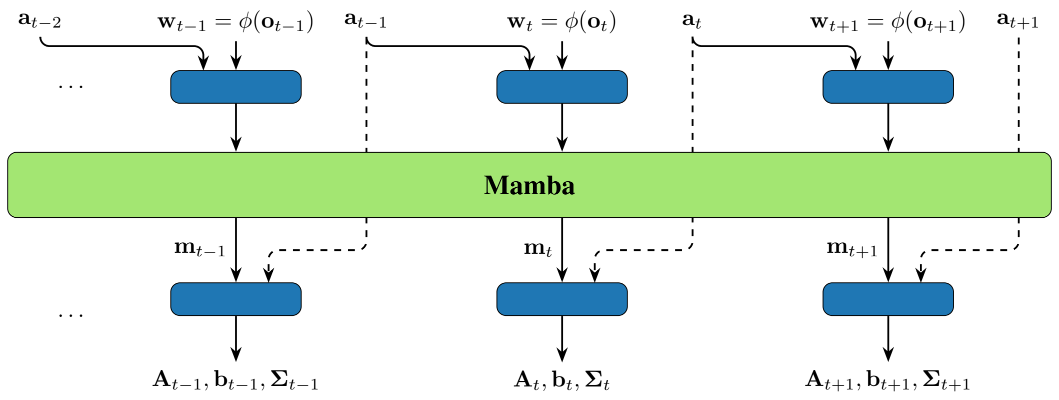
As a remedy, the *Variational Recurrent Kalman Network (VRKN)* [5] builds on a linear Gaussian SSM in a latent space which allows inferring smoothed belief states $\mathbf{q}(\mathbf{z}_t|\mathbf{o}_{\leq T}, \mathbf{a}_{\leq T})$ required for a tight bound. The *VRKN* removes the need for a deterministic path and improves performance under uncertainty. However, it linearizes the dynamics model around the mean of the filtered belief, resulting in a nonlinear autoregressive process that cannot be parallelized.

In contrast, *Recall to Image (R2I)* [41] builds on the *RSSM* and improves computational efficiency at the cost of a more simplistic inference scheme. It uses *S4* [16] instead of a *GRU* to parameterize the deterministic path f but additionally has to remove the inference's dependency on $\mathbf{h}_t$ to allow efficient parallel computation. The resulting inference model, $q(\mathbf{z}_t|\mathbf{o}_t)$ is non-recurrent and neglects all information from other time steps. Thus, while *R2I* excels on memory tasks, it is highly susceptible to noise and partial-observability as the inference cannot account for inconsistent or missing information in $\mathbf{o}_t$.

Our approach, *KalMamba*, combines the tight variational lower bound of the *VRKN* with a parallelizable *Mamba* [15] backbone to learn the parameters of the dynamics. It thus omits the nonlinear autoregressive linearization process. Combined with our custom PyTorch routines for time-parallel filtering and smoothing [42], this approach allows efficient training with the *VRKN*s principled, uncertainty-capturing objective.

4 KalMamba

On a high level, *KalMamba* embeds a linear Gaussian State Space Model into a latent space and learns its dynamics model's parameters using a backbone consisting of several mamba layers. It employs a time-parallel Kalman smoother in this latent space to infer latent beliefs for training and acting. By exploiting the associativity of the underlying operations, we can utilize parallel scans



$\mathbf{a}_{t-2}$
$\mathbf{w}_{t-1} = \phi(\mathbf{o}_{t-1})$
$\mathbf{a}_{t-1}$
$\mathbf{w}_{t} = \phi(\mathbf{o}_{t})$
$\mathbf{a}_{t}$
$\mathbf{w}_{t+1} = \phi(\mathbf{o}_{t+1})$
$\mathbf{a}_{t+1}$
...
Mamba
$\mathbf{m}_{t-1}$
$\mathbf{m}_{t}$
$\mathbf{m}_{t+1}$
...
$\mathbf{A}_{t-1}, \mathbf{b}_{t-1}, \mathbf{\Sigma}_{t-1}$
$\mathbf{A}_{t}, \mathbf{b}_{t}, \mathbf{\Sigma}_{t}$
$\mathbf{A}_{t+1}, \mathbf{b}_{t+1}, \mathbf{\Sigma}_{t+1}$

Figure 2: Schematic of the Mamba [15] based backbone to learn the system dynamics. It shares the inference model's encoder $\phi(\mathbf{o}_t)$ and intermediate representation $\mathbf{w}_t$. Each $\mathbf{w}_t$ is then concatenated to the previous action $\mathbf{a}_{t-1}$, fed through a small Neural Network (NN) and given to *Mamba* model which accumulates information over time and emits a representation $\mathbf{m}_t(\mathbf{o}_{t\leq}, \mathbf{a}_{\leq t-1})$ containing the same information as the filtered belief $q(\mathbf{z}_t|\mathbf{o}_{t\leq}, \mathbf{a}_{\leq t-1})$. We then concatenate each $\mathbf{m}_t$ with the current action $\mathbf{a}_t$ and use another small NN to compute the dynamics parameters $\mathbf{A}_t$, $\mathbf{b}_t$ and $\mathbf{\Sigma}_t$. This scheme allows us to use the intermediate representation $\mathbf{m}_t$ for regularization and we regularize it towards the filtered belief's mean using a Mahalanobis regularizer (c.f. Equation 2). Finally, the small NNs include Monte-Carlo Dropout [14] to model epistemic uncertainty.

for this parallelization. *KalMamba* employs a tight variational lower bound objective that allows appropriate modeling of uncertainties in noisy, partial-observable systems. We then use a *Soft Actor Critic* [21] approach to learn to act, avoiding autoregressive rollouts for policy learning.

4.1 The *KalMamba* Model

To connect the original, high-dimensional observations $\mathbf{o}_t$ to the latent space for inference, we introduce an intermediate auxiliary observation $\mathbf{w}_t$, which is connected to the latent state by an observation model $q(\mathbf{w}_t|\mathbf{z}_t) = \mathcal{N}(\mathbf{w}_t|\mathbf{z}_t, \mathbf{\Sigma}^{\mathbf{w}}_t)$ [6, 45]. Here, we assume $\mathbf{w}_t$ to be observable and extract it, together with the diagonal observation covariance $\mathbf{\Sigma}^{\mathbf{w}}_t$ from the observation using an encoder network; $(\mathbf{w}_t, \mathbf{\Sigma}^{\mathbf{w}}_t) = \phi(\mathbf{o}_t)$. This approach allows us to model the complex dependency between $\mathbf{z}_t$ and $\mathbf{o}_t$ using the encoder while having a simple observation model for inference in the latent space. Opposed to modeling $\mathbf{w}_t$ as a random variable [13, 32], modeling it is observable results in fewer latent variables which simplifies inference and allows direct propagation of the observation uncertainties from the encoder to the state.

We parameterize the dynamics model as

$$p(\mathbf{z}_{t+1}|\mathbf{z}_t, \mathbf{a}_t) = \mathcal{N}\left(\mathbf{z}_{t+t}|\mathbf{A}_t(\mathbf{o}_{\leq t}, \mathbf{a}_{\leq t})\mathbf{z}_t + \mathbf{b}_t(\mathbf{o}_{\leq t}, \mathbf{a}_{\leq t}), \mathbf{\Sigma}^{\text{dyn}}_t(\mathbf{o}_{\leq t}, \mathbf{a}_{\leq t})\right) \tag{1}$$

where both $\mathbf{A}_t$ and $\mathbf{\Sigma}^{\text{dyn}}_t$ are diagonal matrices and we constrain the (eigen)values of $\mathbf{A}$ to be between 0.4 and 0.99. This constraint ensures the resulting dynamics are plausible and stable. This approach effectively linearizes the dynamics parameters $\mathbf{A}_t, \mathbf{b}_t$ and $\mathbf{\Sigma}^{\text{dyn}}_t$ around all past observations and actions. Crucially, the resulting dynamics are linear in $\mathbf{z}_t$ enabling the closed-form inference of beliefs using standard Kalman filtering and smoothing.

Parameterizing the dynamics model of Equation 1 naively can lead to poor representations, as information can bypass the actual SSM through the linearization backbone. To counter this, we design the backbone architecture as depicted in Figure 2. For each timestep, we concatenate $\mathbf{w}_t$ and $\mathbf{a}_{t-1}$, transform each resulting vector using a small neural network, feed it through a *Mamba* [15] model and linearly project the output to a vector $\mathbf{m}_t$ of the same dimension as the latent state $\mathbf{z}_t$. Each $\mathbf{m}_t$ now accumulates the same observations and actions used to form the corresponding filtered belief $q(\mathbf{z}_t|\mathbf{o}_{\leq t}, \mathbf{a}_{\leq t-1})$. We then take $\mathbf{m}_t$ and the action $\mathbf{a}_t$ to compute the dynamics parameters using another small neural network. This bottleneck introduced by $\mathbf{m}_t$ allows us to regularize the model as discussed below. Following [5] we further include Monte-Carlo Dropout [14] into this architecture, as explicitly modeling the epistemic uncertainty is crucial when working with a smoothing inference.

The generative observation model is given by a decoder network $p(\mathbf{o}_t|\mathbf{z}_t)$. The observations are modeled as Gaussian with learned mean and fixed standard deviation. Finally, we assume an initial state distribution $p(\mathbf{z}_0)$ that is a zero mean Gaussian with a learned variance $\mathbf{\Sigma}_0$.

Given the latent observation model $q(\mathbf{w}_t|\mathbf{z}_t)$, and the shared, pre-computable, linear dynamics model, we can efficiently infer belief states using extended Kalman filtering and smoothing. Recent work [42] shows how to formulate such filtering and smoothing as associative operations amenable to temporal parallelization using associative scans. We implement these operations in PyTorch [39]. Similar to *S5* [48] or *Mamba* [15] this implementation yields a logarithmic time complexity, given sufficiently many parallel cores. Additionally, as the dynamics matrix $\mathbf{A}_t$ and all model covariances, i.e., $\mathbf{\Sigma}_t^{\text{dym}}$, $\mathbf{\Sigma}_t^{\text{obs}}$, and $\mathbf{\Sigma}_0$, are diagonal, the same holds for the covariances of the filtered and smoothed beliefs. Thus we can replace costly matrix operations during Kalman filtering and smoothing with point-wise operations, which further ensures *KalMamba's* efficiency.

4.2 Training the Model

After inserting the state space assumptions of our generative and inference models, the standard variational lower bound to the data marginal log-likelihood [31] for a single sequence simplifies to [5]

$$\mathcal{L}_{\text{ssm}}(\mathbf{o}_{\leq T}, \mathbf{a}_{\leq T}) = \sum_{t=1}^{T} \Big(\mathbb{E}_{q(\mathbf{z}_t|\mathbf{o}_{\leq T}, \mathbf{a}_{\leq T})} \left[\log p(\mathbf{o}_t|\mathbf{z}_t) \right] - \\ \mathbb{E}_{q(\mathbf{z}_{t-1}|\mathbf{o}_{\leq T}, \mathbf{a}_{\leq T})} \left[\text{KL} \left[q(\mathbf{z}_t|\mathbf{z}_{t-1}, \mathbf{a}_{\geq t-1}, \mathbf{o}_{\geq t}) \parallel p(\mathbf{z}_t|\mathbf{z}_{t-1}, \mathbf{a}_{t-1}) \right] \right] \Big).$$

Due to the smoothing inference, this lower bound is tight and allows accurate modeling of the underlying system's uncertainties. To evaluate the lower bound we need the smoothed dynamics $q(\mathbf{z}_t|\mathbf{z}_{t-1}, \mathbf{a}_{\geq t-1}, \mathbf{o}_{\geq t})$ whose parameters we can compute given the equations provided in [5].

To regularize the *Mamba*-based backbone used to learn the dynamics, we incentivize $\mathbf{m}_t$ to correspond to the filtered mean using a Mahalonobis distance

$$R(\mathbf{o}_{\leq T}, \mathbf{a}_{\leq T}) = \sum_{t=1}^{T} \left(\mathbf{m}_t(\mathbf{o}_{\leq t}, \mathbf{a}_{\leq t-1} - \boldsymbol{\mu}_t^+ \right)^T \left(\mathbf{\Sigma}_t^+ \right)^{-1} \left(\mathbf{m}_t(\mathbf{o}_{\leq t}, \mathbf{a}_{\leq t-1}) - \boldsymbol{\mu}_t^+ \right), \qquad (2)$$

$\boldsymbol{\mu}_t^+$ and $\mathbf{\Sigma}_t^+$ denote the mean and variance of the filtered belief $q(\mathbf{z}_t|\mathbf{o}_{\leq t}, \mathbf{a}_{\leq t-1})$. This regularization discourages the model from bypassing information over the *Mamba* backbone. This mirrors many established models such as the classical extend Kalman Filter [27], which linearize directly around this mean, but still allows associative parallel scanning.

Finally, we add a reward model $p(r_t|\mathbf{z}_t)$, predicting the current reward from the latent state using a small neural network. While this is not strictly necessary for standard policy learning on top of the representation, it nevertheless helps the model to focus on task-relevant details and learn a good representation for control [49, 51]. Including this reward term and the Mahalonobis regularize, the full maximization objective for a single sequence is given as

$$\mathcal{L}_{\text{KalMamba}}(\mathbf{o}_{\leq T}, \mathbf{a}_{\leq T}) = \mathcal{L}_{\text{ssm}}(\mathbf{o}_{\leq T}, \mathbf{a}_{\leq T}) + \mathbb{E}_{q(\mathbf{z}_t|\mathbf{o}_{\leq T}, \mathbf{a}_{\leq T})} \left[\log p(r_t|\mathbf{z}_t) \right] - \alpha R(\mathbf{o}_{\leq T}, \mathbf{a}_{\leq T}).$$

4.3 Using *KalMamba* for Reinforcement Learning

We learn a policy on top of the *KalMamba* state space representation using *Soft Actor Critic (SAC)* [21]. Here, we use the mean of the variational filtered belief $q(\mathbf{z}_t|\mathbf{o}_{\leq t}, \mathbf{a}_{\leq t-1})$ as input for the actor and, together with the action $\mathbf{a}_t$ for the critic. Importantly, we cannot smooth during acting as future observations and actions are unavailable. However, while not directly involved in the loss, the filter belief is still meaningful as the smoothing pass introduces no additional parameters. This inductive bias induces a tight coupling between filtered and smoothed belief that ensures the reasonableness of the former. We independently train the *KalMamba* world model and *SAC* by stopping the actor's and critic's gradients from propagating through the world model. We use *SAC* instead of the typical latent

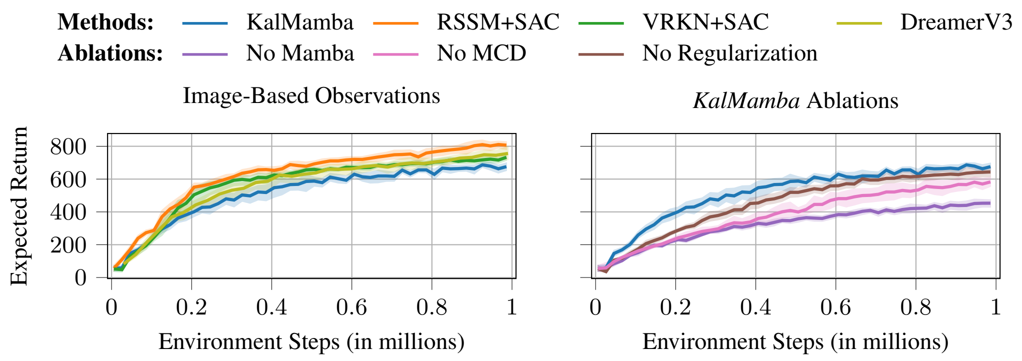


Figure 3: Aggregated expected returns for image-based observations. (**Left**) *KalMamba* is slightly worse but overall competitive with the different baselines. Combining either baseline SSM with SAC matches or exceeds the performance of *DreamerV3*. (**Right**) Using *Mamba* to learn the dynamics is crucial for good model performance. Similarly, both Monte-Carlo Dropout and the regularization loss of Equation 2 stabilize the training process and lead to higher expected returns.

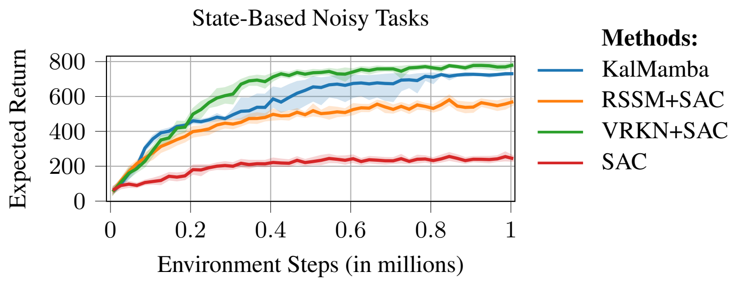


Figure 4: Aggregated expected returns for the state-based noisy tasks. *KalMamba* clearly outperforms the *RSSM* while almost matching the *VRKN*'s performance. Naively using *SAC* is insufficient, which testifies to the increased difficulty due to the noise.

imagination strategy used with *RSSMs*, the *VRKN* and *R2I*. For all 4 models, rolling out policies in the latent space is autoregressive but these rollouts can be avoided by using a Q-function directly on the inferred belief states.

5 Experiments

We evaluate *KalMamba* on 4 tasks from the DeepMind Control (DMC) Suite, namely `cartpule_swingup`, `quardruped_walk`, `walker_walk`, and `walker_run`. We train each task for 1 million environment steps with sequences of length 32 and run 20 evaluation runs every $20,000$ steps. We report the expected return using the mean and 95% stratified bootstrapped confidence intervals [1] for 4 seeds per environment. Appendix A provides all hyperparameters. Appendix B provides per-task results for all experiments.

We compare against *Recurrent State Space Models (RSSM)* and the *Variational Recurrent Kalman Network (VRKN)*. To isolate the effect of the SSMs' representations, we combine both with *SAC* [21] as the RL algorithm, instead of using latent imagination [22].

5.1 Standard Image Based Tasks

We first compare on standard image-based observations of the different tasks and include *DreamerV3* [25] results for reference. The left side of Figure 3 shows the aggregated expected returns. The results indicate that *KalMamba* is slightly worse, but overall competitive to the two baseline SSMs and *DreamerV3*, while being parallelizable and thus much more efficient to train. Interestingly, both SSMs work well when combined with *SAC*, matching or outperforming *DreamerV3*.

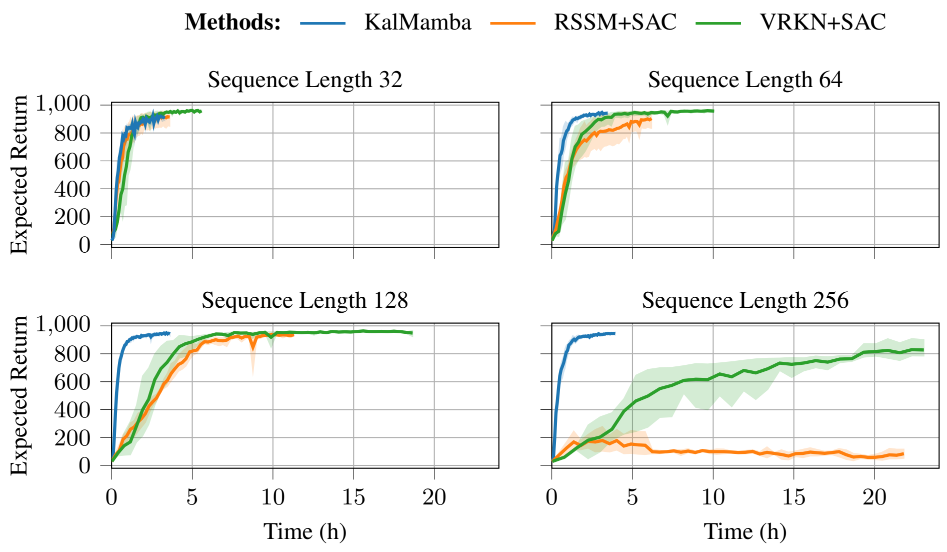


Figure 5: Wall-clock time evaluations on the state-based noisy `walker-walk` for *KalMamba* , the *RSSM*, and the *VRKN* for different training context lengths for 1 million environment steps or up to 24 hours. This time limitation only affected the *VRKN* training for 256 steps, which reached 650 thousand steps after 24 hours. While all methods work well for short sequences of length 32 (**Top Left**), the efficient parallelization of *KalMamba* allows it to scale gracefully to and even improve performance for longer sequences of up to 256 steps, where the other methods fail (**Bottom Right**).

We also conduct ablations for some of the main design choices of *KalMamba* on the right side of Figure 3. *No Mamba* removes the Mamba layers from the dynamics backbone in Figure 2. Similar to the selection mechanism of *Mamba* [15] itself, the resulting approach linearizes the dynamics around the current action and observation, instead of all previous observations and actions. The results show that this is insufficient for *KalMamba*, presumably because it uses only a single SSM layer instead of the stacked layers used by *Mamba*. Furthermore, *No Regularization loss* removes the Mahalanobis regularization from the model and *No Monte Carlo Dropout* removes Monte-Carlo Dropout from the dynamics backbone. Here, the results indicate that regularizing $\mathbf{m}_t$ and explicitly modeling the epistemic uncertainty are crucial for *KalMamba's* performance.

5.2 Low Dimensional Tasks with Observation and Dynamics Noise

To test the models' capabilities under uncertainties, we use the state-based versions of the tasks and add both observation and dynamics noise. The observation noise is sampled from $\mathcal{N}(0, 0.3)$ and added to the observation. The dynamics noise is also sampled from $\mathcal{N}(0, 0.3)$ and added to the action before execution. However, unlike exploration noise, this addition happens inside the environment and is invisible to the world model and the policy. We include *SAC* without a world model in our experiments as a baseline to evaluate the difficulty of the resulting tasks.

The results in Figure 4 show that naively using *SAC* fails in the presence of noise. While the *RSSM* manages to improve performance it is still significantly outperformed by *VRKN* and *KalMamba*, which both use the robust smoothing inference scheme. *KalMamba* needs slightly more environment steps to converge but ultimately almost matches the *VRKN*'s performance while being significantly faster to run.

5.3 Runtime Analysis

To show the benefit of *KalMambas* efficient parallelization using associative scans, we compare its wall-clock runtime to that of the SSM baselines on the state-based noisy version of `walker-walk` for training sequences of increasing length. The models share a PyTorch implementation and differ only in the SSM. We run each experiment on a single Nvidia Tesla H100 GPU, for up to 1 million steps or 24 hours. Figure 5 shows the resulting expected returns. While all models work well

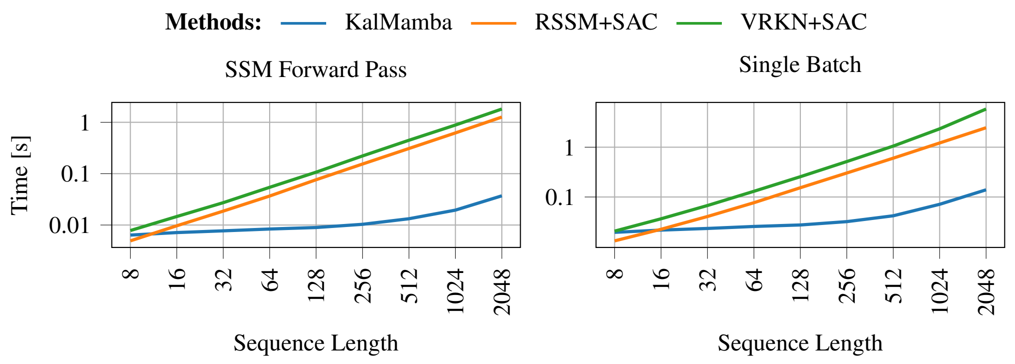


Figure 6: Runtime comparison of *KalMamba*, the RSSM and the VRKN for (**Left**) a SSM forward pass and (**Right**) a single training batch. While the computational cost of both baseline models scales linearly in the sequence length, *KalMamba* utilizes associative scans for efficient parallelism and thus near-logarithmic runtime on modern accelerator hardware.

for the short sequences of length 32 that are used for the main results above, the training time of the baseline SSMs scales linearly with the sequence length, causing slower convergence and a time-out after 24 hours and 650 thousand environment steps for the *VRKN* for a length of 256. In comparison, *KalMamba* shows negligible additional training cost for increased sequence lengths. Further, while the absolute performance of both baselines decreases as the training sequences get longer, *KalMamba* slightly improves performance when trained on more than 32 steps. These results indicate that *KalMamba* efficiently utilizes long-term context information through its *Mamba* backbone, whereas the dynamics models of the baseline SSMs have difficulty with too-long training sequences.

Investigating this further, we visualize the wall-clock time of a single SSM forward pass and a single training batch for different sequence lengths in Figure 6. While both the *RSSM* and *VRKN* scale linearly with the sequence length, *KalMamba* shows near-logarithmic scaling even for longer sequences thanks to its efficient parallelism. We expect further significant speedups for *KalMamba* with a potential custom CUDA implementation, similar to *Mamba*.

6 Conclusion

We proposed *KalMamba*, an efficient State Space Model (SSM) for Reinforcement Learning (RL) under uncertainty. It combines the uncertainty awareness of probabilistic SSMs with recent deterministic SSMs' scalability by embedding a linear Gaussian SSM into a latent space. We use *Mamba* [15] to learn the linearized dynamics in this latent space efficiently. Inference in this SSM amounts to standard Kalman filtering and smoothing and is amenable to full parallelization using associative scans [42]. During model learning, this allows time-parallel estimation of smoothed belief states, which allows the efficient usage of principled objectives for uncertainty estimation, especially over long sequences.

Our experiments on low-dimensional states and image observations indicate that *KalMamba* can match the performance of state-of-the-art stochastic SSMs for RL under uncertainty. In terms of both runtime and performance, *KalMamba* scales more gracefully to longer training sequences. In particular, its performance improves with sequence length while it degrades for the baseline SSMs.

Limitations and Future Work. The present work explores *KalMamba*'s potential in small-scale experiments, but a more elaborate evaluation of diverse, more realistic tasks would help to explore our method's strengths and weaknesses. A thorough comparison of recent baselines is needed to contextualize *KalMamba* against existing time-efficient SSMs with simplified, non-smoothing inference schemes [41]. We additionally aim to refine *KalMamba* to improve its performance across wide-ranging tasks. In this context, we plan to model the state with a complex-valued random variable to expand the range of dynamics models that can be learned. Other ideas include improving the regularization of the Mamba backbone and investigating more advanced policy learning methods that make use of the uncertainty in the filtered beliefs.

References

[1] R. Agarwal, M. Schwarzer, P. S. Castro, A. C. Courville, and M. Bellemare. Deep reinforcement learning at the edge of the statistical precipice. *Advances in neural information processing systems*, 34:29304–29320, 2021.

[2] E. Archer, I. M. Park, L. Buesing, J. Cunningham, and L. Paninski. Black box variational inference for state space models. *arXiv preprint arXiv:1511.07367*, 2015.

[3] E. Banijamali, R. Shu, H. Bui, A. Ghodsi, et al. Robust locally-linear controllable embedding. In *International Conference on Artificial Intelligence and Statistics*, pages 1751–1759. PMLR, 2018.

[4] P. Becker, S. Markgraf, F. Otto, and G. Neumann. Joint representations for reinforcement learning with multiple sensors. *arXiv preprint arXiv:2302.05342*, 2023.

[5] P. Becker and G. Neumann. On uncertainty in deep state space models for model-based reinforcement learning. *Transactions on Machine Learning Research*, 2022.

[6] P. Becker, H. Pandya, G. Gebhardt, C. Zhao, C. J. Taylor, and G. Neumann. Recurrent kalman networks: Factorized inference in high-dimensional deep feature spaces. In *International Conference on Machine Learning*, pages 544–552, 2019.

[7] P. Becker-Ehmck, J. Peters, and P. Van Der Smagt. Switching linear dynamics for variational Bayes filtering. In K. Chaudhuri and R. Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 553–562. PMLR, 09–15 Jun 2019.

[8] C. Chen, Y.-F. Wu, J. Yoon, and S. Ahn. Transdreamer: Reinforcement learning with transformer world models. *arXiv preprint arXiv:2202.09481*, 2022.

[9] K. Cho, B. Van Merriënboer, D. Bahdanau, and Y. Bengio. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*, 2014.

[10] K. Chua, R. Calandra, R. McAllister, and S. Levine. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. *Advances in neural information processing systems*, 31, 2018.

[11] A. Doerr, C. Daniel, M. Schiegg, N.-T. Duy, S. Schaal, M. Toussaint, and T. Sebastian. Probabilistic recurrent state-space models. In *International Conference on Machine Learning*, pages 1280–1289. PMLR, 2018.

[12] S. Eleftheriadis, T. Nicholson, M. P. Deisenroth, and J. Hensman. Identification of gaussian process state space models. In *NIPS*, pages 5309–5319, 2017.

[13] M. Fraccaro, S. Kamronn, U. Paquet, and O. Winther. A disentangled recognition and nonlinear dynamics model for unsupervised learning. In *Advances in Neural Information Processing Systems*, pages 3601–3610, 2017.

[14] Y. Gal and Z. Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pages 1050–1059. PMLR, 2016.

[15] A. Gu and T. Dao. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*, 2023.

[16] A. Gu, K. Goel, and C. Re. Efficiently modeling long sequences with structured state spaces. In *International Conference on Learning Representations*, 2021.

[17] S. Gu, Z. Ghahramani, and R. Turner. Neural adaptive sequential monte carlo. *Advances in Neural Information Processing Systems*, 2015:2629–2637, 2015.

[18] A. Gupta, A. Gu, and J. Berant. Diagonal state spaces are as effective as structured state spaces. *Advances in Neural Information Processing Systems*, 35:22982–22994, 2022.

[19] D. Ha and J. Schmidhuber. World models. *arXiv preprint arXiv:1803.10122*, 2018.

[20] T. Haarnoja, A. Ajay, S. Levine, and P. Abbeel. Backprop kf: learning discriminative deterministic state estimators. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, pages 4383–4391, 2016.

[21] T. Haarnoja, A. Zhou, K. Hartikainen, G. Tucker, S. Ha, J. Tan, V. Kumar, H. Zhu, A. Gupta, P. Abbeel, et al. Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*, 2018.

[22] D. Hafner, T. Lillicrap, J. Ba, and M. Norouzi. Dream to control: Learning behaviors by latent imagination. In *International Conference on Learning Representations*, 2020.

[23] D. Hafner, T. Lillicrap, I. Fischer, R. Villegas, D. Ha, H. Lee, and J. Davidson. Learning latent dynamics for planning from pixels. In *International conference on machine learning*, pages 2555–2565. PMLR, 2019.

[24] D. Hafner, T. P. Lillicrap, M. Norouzi, and J. Ba. Mastering atari with discrete world models. In *International Conference on Learning Representations*, 2021.

[25] D. Hafner, J. Pasukonis, J. Ba, and T. Lillicrap. Mastering diverse domains through world models. *arXiv preprint arXiv:2301.04104*, 2023.

[26] R. Hasani, M. Lechner, T.-H. Wang, M. Chahine, A. Amini, and D. Rus. Liquid structural state-space models. In *The Eleventh International Conference on Learning Representations*, 2022.

[27] A. Jazwinski. *Stochastic processes and filtering theory*. ACADEMIC PRESS, INC.,, 1970.

[28] R. E. Kalman. A new approach to linear filtering and prediction problems. *Journal of basic Engineering*, 82(1):35–45, 1960.

[29] M. Karl, M. Soelch, J. Bayer, and P. van der Smagt. Deep variational bayes filters: Unsupervised learning of state space models from raw data. *arXiv preprint arXiv:1605.06432*, 2016.

[30] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

[31] D. P. Kingma and M. Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[32] A. Klushyn, R. Kurle, M. Soelch, B. Cseke, and P. van der Smagt. Latent matters: Learning deep state-space models. *Advances in Neural Information Processing Systems*, 34, 2021.

[33] R. Krishnan, U. Shalit, and D. Sontag. Structured inference networks for nonlinear state space models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31, 2017.

[34] A. X. Lee, A. Nagabandi, P. Abbeel, and S. Levine. Stochastic latent actor-critic: Deep reinforcement learning with a latent variable model. *Advances in Neural Information Processing Systems*, 33:741–752, 2020.

[35] A. Moretti, Z. Wang, L. Wu, and I. Pe'er. Smoothing nonlinear variational objectives with sequential monte carlo, 2019.

[36] K. P. Murphy. *Machine learning: a probabilistic perspective*. MIT press, 2012.

[37] C. Naesseth, S. Linderman, R. Ranganath, and D. Blei. Variational sequential monte carlo. In *International Conference on Artificial Intelligence and Statistics*, pages 968–977. PMLR, 2018.

[38] T. D. Nguyen, R. Shu, T. Pham, H. Bui, and S. Ermon. Temporal predictive coding for model-based planning in latent space. In *International Conference on Machine Learning*, pages 8130–8139. PMLR, 2021.

[39] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

[40] H. E. Rauch, F. Tung, and C. T. Striebel. Maximum likelihood estimates of linear dynamic systems. *AIAA journal*, 3(8):1445–1450, 1965.

[41] M. R. Samsami, A. Zholus, J. Rajendran, and S. Chandar. Mastering memory tasks with world models. In *The Twelfth International Conference on Learning Representations*, 2024.

[42] S. Särkkä and Á. F. García-Fernández. Temporal parallelization of bayesian smoothers. *IEEE Transactions on Automatic Control*, 66(1):299–306, 2020.

[43] F. Schmidt and T. Hofmann. Deep state space models for unconditional word generation. *Advances in Neural Information Processing Systems 31*, 31:6158–6168, 2018.

[44] S. Sengupta, M. Harris, Y. Zhang, and J. D. Owens. Scan primitives for gpu computing, 2007.

[45] V. Shaj, P. Becker, D. Buchler, H. Pandya, N. van Duijkeren, C. J. Taylor, M. Hanheide, and G. Neumann. Action-conditional recurrent kalman networks for forward and inverse dynamics learning. *Conference on Robot Learning*, 2020.

[46] V. Shaj, D. Büchler, R. Sonker, P. Becker, and G. Neumann. Hidden parameter recurrent state space models for changing dynamics scenarios. In *International Conference on Learning Representations*, 2022.

[47] R. H. Shumway and D. S. Stoffer. An approach to time series smoothing and forecasting using the em algorithm. *Journal of time series analysis*, 3(4):253–264, 1982.

[48] J. T. Smith, A. Warrington, and S. Linderman. Simplified state space layers for sequence modeling. In *The Eleventh International Conference on Learning Representations*, 2022.

[49] N. Srivastava, W. Talbott, M. B. Lopez, S. Zhai, and J. M. Susskind. Robust robotic control from pixels using contrastive recurrent state-space models. In *Deep RL Workshop NeurIPS 2021*, 2021.

[50] Y. Tassa, Y. Doron, A. Muldal, T. Erez, Y. Li, D. d. L. Casas, D. Budden, A. Abdolmaleki, J. Merel, A. Lefrancq, et al. Deepmind control suite. *arXiv preprint arXiv:1801.00690*, 2018.

[51] M. Tomar, U. A. Mishra, A. Zhang, and M. E. Taylor. Learning representations for pixel-based control: What matters and why? *Transactions on Machine Learning Research*, 2023.

[52] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[53] M. Watter, J. Springenberg, J. Boedecker, and M. Riedmiller. Embed to control: A locally linear latent dynamics model for control from raw images. In *Advances in neural information processing systems*, pages 2746–2754, 2015.

[54] P. Wu, A. Escontrela, D. Hafner, P. Abbeel, and K. Goldberg. Daydreamer: World models for physical robot learning. In *6th Annual Conference on Robot Learning*, 2022.

[55] L. Yingzhen and S. Mandt. Disentangled sequential autoencoder. In J. Dy and A. Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 5670–5679. PMLR, 10–15 Jul 2018.

[56] L. Zhou, M. Poli, W. Xu, S. Massaroli, and S. Ermon. Deep latent state space models for time-series generation. In *International Conference on Machine Learning*, pages 42625–42643. PMLR, 2023.

A Hyperparameters and Implementation Details

Table 2 lists all hyperparameters of the *KalMamba* model and Table 3 lists the hyperparameters of *Soft Actor Critic (SAC)* [21] used for control.

Table 2: World Model Hyperparameters

Hyperparameter	Low Dimensional DMC	Image Based DMC
	World Model	
Encoder	2×256 Unit NN with ELU	ConvNet from [19, 22] with ReLU
Decoder	2×256 Unit NN with ELU	ConvNet from [19, 22] with ReLU
Reward Decoder	2×256 Unit NN with ELU	
Latent Space Size	230 (30 Stoch. + 200 Det. for RSSM	
	Mamba Backbone	
num blocks	2	
d_model	256	
d_state	64	
d_conv	2	
dropout probability	0.1	
activation	SiLU	
pre mamba layers	2×256 Unit NN with SiLU	
post mamba layers	VRKN Dynamics Model Architecture from [5] with SiLU	
	Loss	
KL Balancing	0.8 for RSSM, 0.5 for VRKN, KalMamba	
Free Nats	3	
α (regularization scale)	1, KalMamba only	
	Optimizer (Adam [30])	
Learning Rate	$3 \cdot 10^{-4}$	

Table 3: SAC Hyperparameters

Hyperparameter	Low Dimensional DMC	Image Based DMC
Actor-Network	2×256 Unit NN with ReLU	3×1024 Unit NN with ELU
Critic-Network	2×256 Unit NN with ReLU	3×1024 Unit NN with ELU
Actor Optimizer	Adam with learning rate 3×10^{-4}	
Critic Optimizer	Adam with learning rate 3×10^{-4}	
Target Critic Update Fraction	0.005	
Target Critic Update Interval	1	
Target Entropy	$-d_{\text{action}}$	
Entropy Optimizer	Adam with learning rate 3×10^{-4}	
Initial Learning Rate	0.1	
discount γ	0.99	

A.1 Baselines.

Both *RSSM+SAC* and *VRKN+SAC* use the same hyperparameters as *KalMamba* where applicable. For all other hyperparameters, we use the defaults from [22] and [5] respectively. The *SAC* baseline uses the hyperparameters listed in Table 3 and the results for *DreamerV3* [25] are provided by the authors[2].

[2] `https://github.com/danijar/dreamerv3`

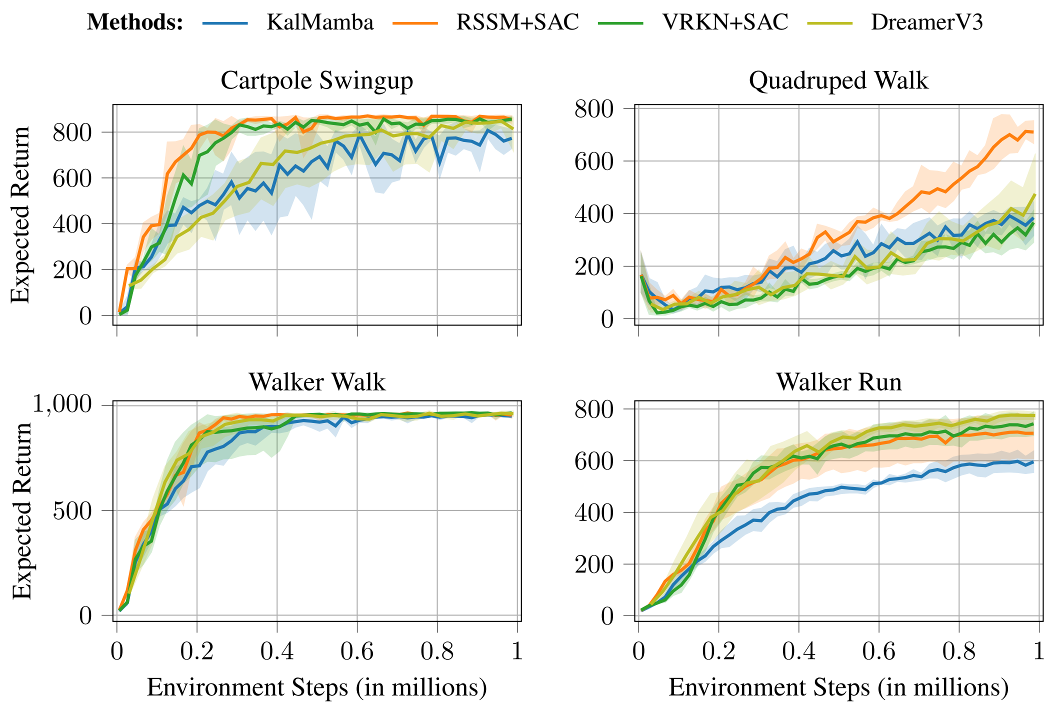


Figure 7: Task-wise evaluations of the DeepMind Control Suite on image-based observations. Dreamer-v3 shows a performance similar to RSSM+SAC.

B Additional Results

We provide results for the individual tasks of the Deepmind Control Suite for image-based observations in Figure 7 and the different *KalMamba* ablations in Figure 8. Figure 9, shows the per-task results for the noisy state-based environments.

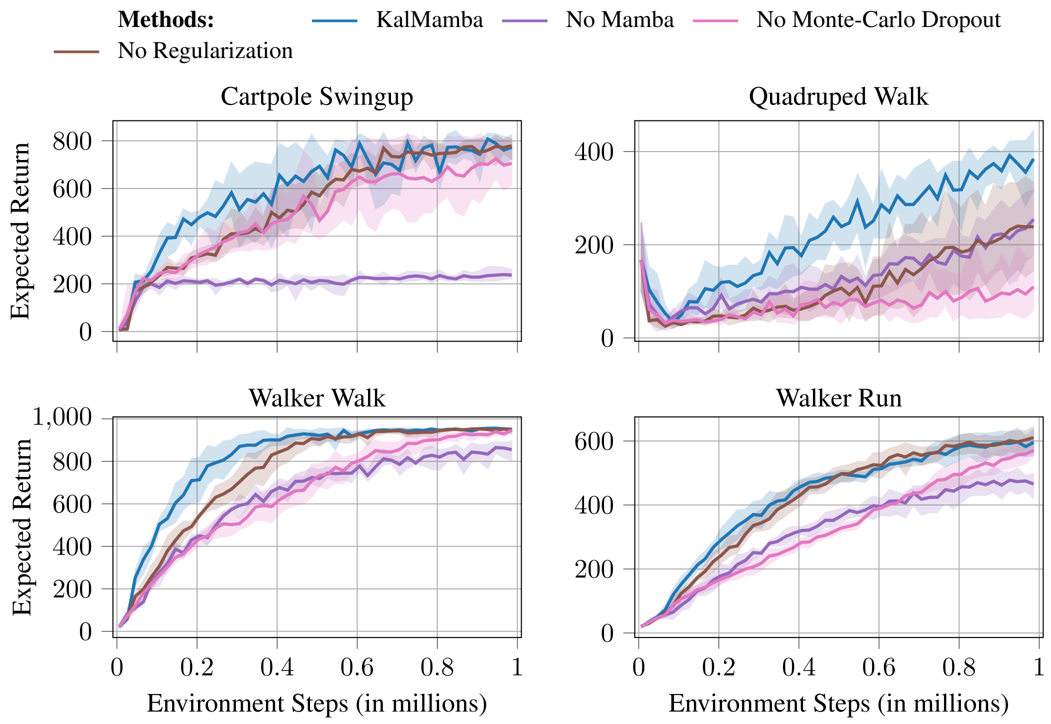


Figure 8: Task-wise evaluations of the DeepMind Control Suite for different *KalMamba* ablations. Monte-Carlo Dropout and the Mahalanobis regularization make the largest difference for the hardest task in the suite, i.e., `quadruped_walk`.

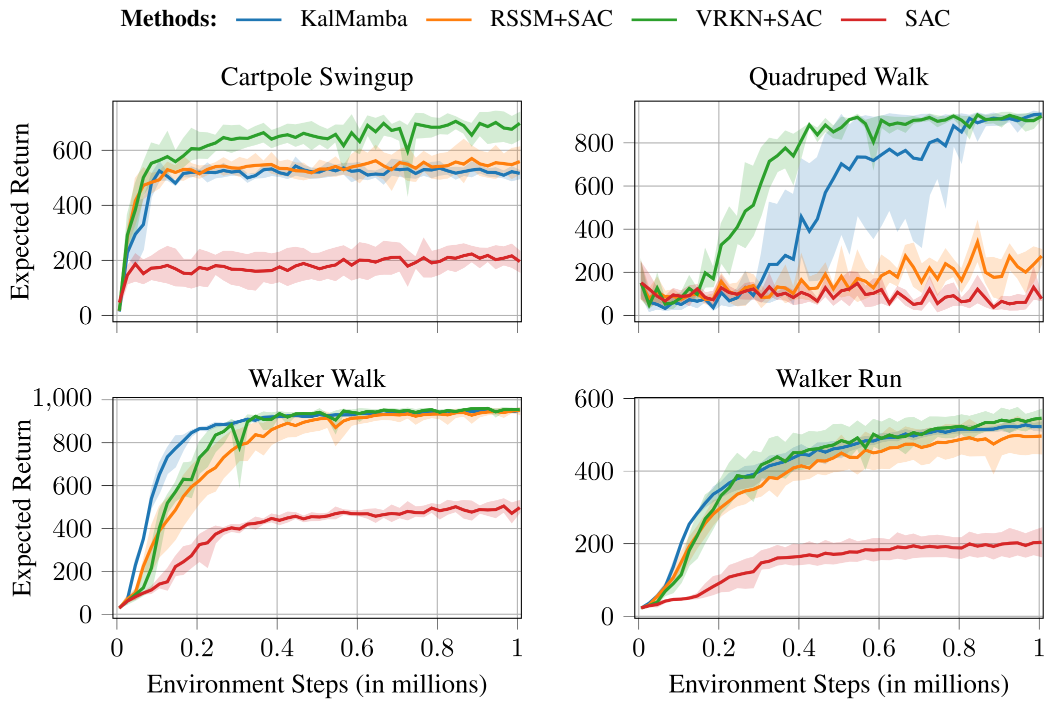


Figure 9: Task-wise evaluations of the DeepMind Control Suite on low-dimensional state representations. *KalMamba* performs on par with or better than the RSSM on all tasks, and is only outperformed by the computationally more expensive VRKN on `cartpole_ swingup`.