Slot State Space Models

Jindong Jiang* Rutgers University
Fei Deng Rutgers University
Gautam Singh Rutgers University
Minseung Lee KAIST
Sungjin Ahn* KAIST

Abstract

Recent State Space Models (SSMs) such as S4, S5, and Mamba have shown remarkable computational benefits in long-range temporal dependency modeling. However, in many sequence modeling problems, the underlying process is inherently modular and it is of interest to have inductive biases that mimic this modular structure. In this paper, we introduce SlotSSMs, a novel framework for incorporating independent mechanisms into SSMs to preserve or encourage separation of information. Unlike conventional SSMs that maintain a monolithic state vector, SlotSSMs maintains the state as a collection of multiple vectors called slots. Crucially, the state transitions are performed independently per slot with sparse interactions across slots implemented via the bottleneck of self-attention. In experiments, we evaluate our model in object-centric learning, 3D visual reasoning, and long-context video understanding tasks, which involve modeling multiple objects and their long-range temporal dependencies. We find that our proposed design offers substantial performance gains over existing sequence modeling methods. Project page is available at `https://slotssms.github.io/`

1 Introduction

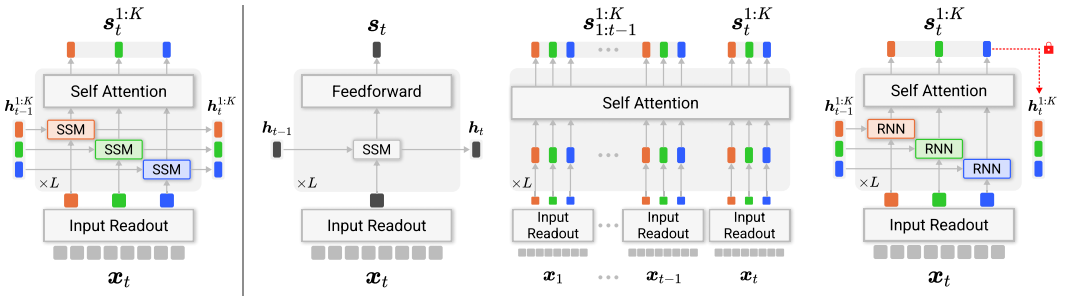
State space models (SSMs) have recently emerged as a promising class of sequence models, achieving remarkable success in language modeling [27, 58, 24, 50, 9] due to their long-term memory capability and computational efficiency. Compared to Transformers [4] whose attention mechanisms also facilitate capturing long-range dependencies, SSMs are more efficient during both training and inference. Notably, SSMs offer parallel training with sub-quadratic complexity, and recurrent generation with constant cost per time step. These benefits have motivated the application of SSMs to sequences of other modalities such as audio [19] and video [10].

Typically, SSMs use a monolithic state vector to summarize all past information. This design can struggle to model sequences with modular underlying structures, which are common in physical processes and real-world dynamics. For example, physical objects largely follow independent dynamics based on their own properties, with strong interactions happening only sparsely (*e.g.*, when objects come in close contact). A monolithic state vector would excessively entangle the dynamics of different entities, thereby hurting generalization. It could be beneficial to incorporate inductive biases for independent mechanisms [20] into the sequence modeling architecture.

Recent progress in object-centric learning [46, 54, 36] has led to several methods for discovering modular object-centric structures and modeling their dynamics from videos with no or only weak supervision [39, 13, 57]. Similar to RIMs [20], they build modularity into the RNN architecture to separately keep track of the dynamics of each object. However, RNNs are prone to vanishing

*Correspondence to `jindong.jiang@rutgers.edu` and `sungjin.ahn@kaist.ac.kr`.

38th Conference on Neural Information Processing Systems (NeurIPS 2024).



(a) SlotSSMs (Ours) (b) Single State SSMs (c) Transformer-based Models (d) RNN-based Models

Figure 1: SlotSSMs vs existing models. (a) SlotSSMs incorporate modularity through independent state transitions and sparse interactions via self-attention. (b) Traditional SSMs utilize a monolithic state vector for all past information. (c) Multi-slot Transformer-based models offer modularity but with high computational complexity. (d) Multi-slot RNN-based models have modular states but can't parallelize training (red lock). SlotSSMs combine parallelizable training, memory efficiency, and modularity for efficient temporal modeling.

gradients [51] and are not amenable to parallel training, making it hard to scale these methods up to modeling long-range effects that span hundreds of time steps.

In this paper, we propose Slot State Space Models (SlotSSMs), a novel and general SSM framework that have built-in inductive biases for discovering and maintaining independent mechanisms. Instead of using monolithic state vectors, SlotSSMs maintain a set of modular slot states whose transition dynamics are designed to be largely independent, with only sparse interaction across slots introduced through the bottleneck of self-attention. The number of slots can be flexible across the layers of SlotSSMs, allowing slots to have a different level of abstraction at each layer. Furthermore, SlotSSMs inherit the strengths of SSMs, namely parallelizable training, memory efficiency, and long-range reasoning capabilities, giving it an advantage over methods based on RNNs and Transformers.

Our contributions are summarized as follows. First, we propose SlotSSMs, a novel and general architecture that incorporates independent mechanisms into SSMs for modeling inherently modular physical processes. Second, we show that SlotSSMs can be specialized to solve object-centric learning tasks. It achieves comparable or better performance than existing RNN-based methods and the Transformer baseline that we develop, while being more computationallly efficient. Third, we further investigate the abilities of SlotSSMs as a general sequence modeling framework, demonstrating its advantages in video understanding and prediction, long-range reasoning, and 3D visual reasoning.

2 Preliminaries

A state space model (SSM) defines a mapping between an input sequence $\boldsymbol{e}_{1:T} \in \mathbb{R}^{T\times D}$ and an output sequence $\boldsymbol{y}_{1:T} \in \mathbb{R}^{T\times D}$ via the recurrence [28, 27, 58, 50]:

$$\begin{aligned} \boldsymbol{h}_t &= \overline{\boldsymbol{A}}_t \boldsymbol{h}_{t-1} + \overline{\boldsymbol{B}}_t \boldsymbol{e}_t \,, \\ \boldsymbol{y}_t &= \boldsymbol{C}_t \boldsymbol{h}_t \,. \end{aligned} \tag{1}$$

Here, T is the sequence length; $\boldsymbol{e}_t, \boldsymbol{y}_t \in \mathbb{R}^D$ are input and output vectors at time t; and $\boldsymbol{h}_t \in \mathbb{R}^H$ is the hidden state summarizing the history $\boldsymbol{e}_{\leq t}$. The matrices $\overline{\boldsymbol{A}}_t \in \mathbb{R}^{H\times H}$, $\overline{\boldsymbol{B}}_t \in \mathbb{R}^{H\times D}$, and $\boldsymbol{C}_t \in \mathbb{R}^{D\times H}$ are designed with learnable parameters in specific ways that encourage modeling long-range dependencies while maintaining computational efficiency. For example, $\overline{\boldsymbol{A}}_t$ commonly takes a diagonal or block-diagonal form, with its (complex) eigenvalues distributed close to the unit circle at initialization [25, 27, 29, 26, 58, 50].

When $\overline{\boldsymbol{A}}_t, \overline{\boldsymbol{B}}_t, \boldsymbol{C}_t$ are time-invariant (constant over t), the computation of $\boldsymbol{y}_{1:T}$ can be parallelized, enabling efficient training. Recent works [24, 9] further show that conditioning these matrices on the input $\boldsymbol{e}_t$ does not hinder training efficiency. They employ learnable functions $\overline{\boldsymbol{A}}\colon \mathbb{R}^D \to \mathbb{R}^{H\times H}, \overline{\boldsymbol{B}}\colon \mathbb{R}^D \to \mathbb{R}^{H\times D}, \boldsymbol{C}\colon \mathbb{R}^D \to \mathbb{R}^{D\times H}$ to generate input-dependent matrices:

$$\overline{\boldsymbol{A}}_t = \overline{\boldsymbol{A}}(\boldsymbol{e}_t) \,, \quad \overline{\boldsymbol{B}}_t = \overline{\boldsymbol{B}}(\boldsymbol{e}_t) \,, \quad \boldsymbol{C}_t = \boldsymbol{C}(\boldsymbol{e}_t) \,. \tag{2}$$

This allows the model to selectively emphasize or ignore information based on the input, leading to more flexible sequence modeling.

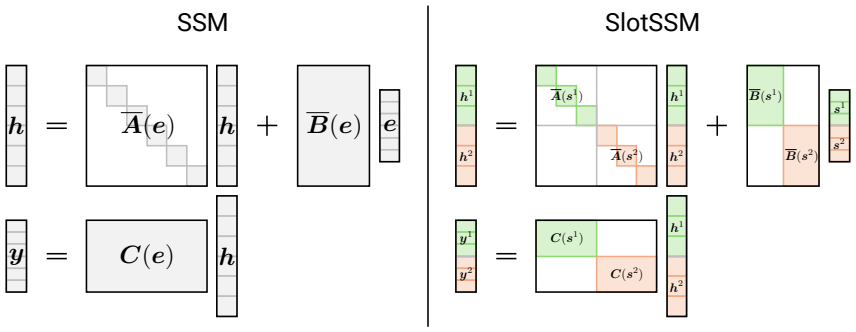


Figure 2: SSM vs SlotSSM. SlotSSM encourages modularity by maintaining a set of separate slot state representations, each updated independently using separate transition matrices and input matrices, allowing for more efficient and scalable modeling of complex sequences with inherent modular structures.

Due to the (block-)diagonal structure of $\overline{\boldsymbol{A}}_t$ limiting cross-dimensional information flow, SSMs are typically interleaved with mixing layers (e.g., linear projections or MLPs) to mix information across dimensions. Alternatively, using dense $\overline{\boldsymbol{B}}_t$ and $\boldsymbol{C}_t$ matrices can also enhance mixing.

3 Slot State Space Models (SlotSSMs)

Standard SSMs use monolithic vectors for inputs, outputs, and hidden states, and mix information across all dimensions. This lack of modularity could cause difficulties in modeling real-world dynamics such as object interactions, where the underlying process consists of multiple entities and is inherently modular [20]. In this section, we present slot state space models (SlotSSMs), a new class of SSMs with built-in inductive biases for encouraging and preserving modularity.

Our key idea is to maintain a set of separate *slot state* representations (called slots in short), and process the slots independently and symmetrically. To do this, we format the input vector $\boldsymbol{e}_t \in \mathbb{R}^D$ as a concatenation of K slot representations $\{\boldsymbol{s}_t^k \in \mathbb{R}^{D_s}\}_{k=1}^K$, where $D_s = D/K$. The output $\boldsymbol{y}_t \in \mathbb{R}^D$ and hidden state $\boldsymbol{h}_t \in \mathbb{R}^H$ are formatted similarly:

$$\boldsymbol{e}_t = \texttt{concat}\left[\boldsymbol{s}_t^1, \ldots, \boldsymbol{s}_t^K\right], \quad \boldsymbol{y}_t = \texttt{concat}\left[\boldsymbol{y}_t^1, \ldots, \boldsymbol{y}_t^K\right], \quad \boldsymbol{h}_t = \texttt{concat}\left[\boldsymbol{h}_t^1, \ldots, \boldsymbol{h}_t^K\right], \quad (3)$$

where $\boldsymbol{y}_t^k \in \mathbb{R}^{D_s}$ and $\boldsymbol{h}_t^k \in \mathbb{R}^{H_s}$ are the output and the hidden state corresponding to slot $\boldsymbol{s}_t^k$, with $H_s = H/K$. In this section, we focus on preserving modularity when the input already complies with the slot format. When coupled with a slot encoder, the SlotSSM can help encourage the emergence of modularity from unstructured inputs such as video frames, as we will discuss in Section 4.

To preserve modularity, we make sure that SlotSSM do not mix information across different slots. More precisely, the hidden state $\boldsymbol{h}_t^k$ and output $\boldsymbol{y}_t^k$ only integrate information from the history of the corresponding input slot $\boldsymbol{s}_{\leq t}^k$. As illustrated in Figure 2 (Right), this can be achieved by making $\overline{\boldsymbol{A}}_t, \overline{\boldsymbol{B}}_t, \boldsymbol{C}_t$ block-diagonal, where the k-th block is only conditioned on the k-th slot:

$$\overline{\boldsymbol{A}}_t = \operatorname{diag}\left(\{\overline{\boldsymbol{A}}(\boldsymbol{s}_t^k)\}_{k=1}^K\right), \quad \overline{\boldsymbol{B}}_t = \operatorname{diag}\left(\{\overline{\boldsymbol{B}}(\boldsymbol{s}_t^k)\}_{k=1}^K\right), \quad \boldsymbol{C}_t = \operatorname{diag}\left(\{\boldsymbol{C}(\boldsymbol{s}_t^k)\}_{k=1}^K\right). \quad (4)$$

Implementation details. The SlotSSM formulation in Equation 4 is general and can accommodate various choices of the $\overline{\boldsymbol{A}}, \overline{\boldsymbol{B}}, \boldsymbol{C}$ functions. In our implementation, we adopt those from Mamba [24]. Specifically, $\overline{\boldsymbol{A}}(\boldsymbol{s}_t^k), \overline{\boldsymbol{B}}(\boldsymbol{s}_t^k), \boldsymbol{C}(\boldsymbol{s}_t^k)$ are themselves block-diagonal matrices with D_s blocks, one for each slot dimension. The i-th blocks $\overline{\boldsymbol{A}}^{(i)}(\boldsymbol{s}_t^k) \in \mathbb{R}^{N\times N}$ and $\overline{\boldsymbol{B}}^{(i)}(\boldsymbol{s}_t^k) \in \mathbb{R}^{N\times 1}$ are obtained by discretizing their continuous-time counterparts $\boldsymbol{A}^{(i)}$ and $\boldsymbol{B}^{(i)}(\boldsymbol{s}_t^k)$ using the time step $\Delta^{(i)}(\boldsymbol{s}_t^k)$ and the zero-order hold (ZOH) rule:

$$\overline{\boldsymbol{A}}^{(i)}(\boldsymbol{s}_t^k),\ \overline{\boldsymbol{B}}^{(i)}(\boldsymbol{s}_t^k) = \operatorname{ZOH}\left(\Delta^{(i)}(\boldsymbol{s}_t^k),\ \boldsymbol{A}^{(i)},\ \boldsymbol{B}^{(i)}(\boldsymbol{s}_t^k)\right), \quad i = 1, \ldots, D_s\,. \quad (5)$$

Here, $N = H_s/D_s$ is the hidden state size per slot dimension, $\boldsymbol{A}^{(i)} \in \mathbb{R}^{N\times N}$ is an input-independent learnable model parameter, and $\Delta^{(i)}\colon \mathbb{R}^D \to \mathbb{R}$, $\boldsymbol{B}^{(i)}\colon \mathbb{R}^D \to \mathbb{R}^{N\times 1}$ are learnable functions implemented as neural networks. Similarly, the i-th block $\boldsymbol{C}^{(i)}(\boldsymbol{s}_t^k)$ is computed by the learnable function

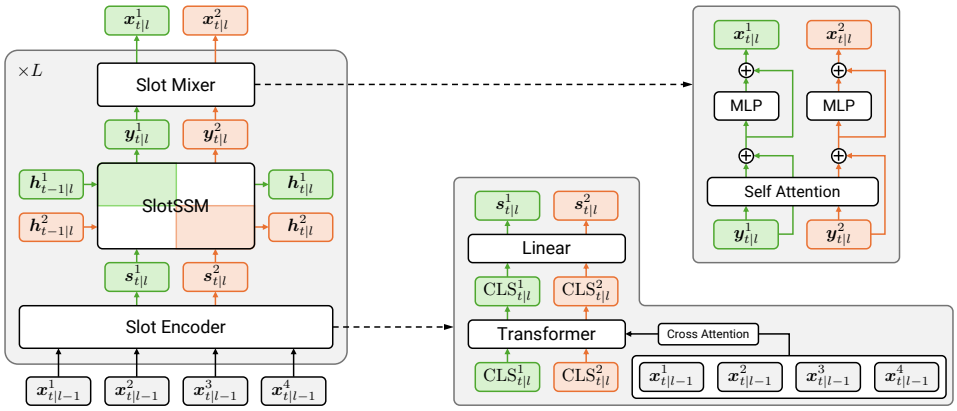


Figure 3: Sequence modeling with SlotSSM. Each layer includes a Slot Encoder, SlotSSM, and Slot Mixer. The Slot Encoder uses a Transformer to extract slots from inputs. The SlotSSM independently updates the slots via separate state transitions. The Slot Mixer introduces inter-slot interactions through self-attention.

$\boldsymbol{C}^{(i)}: \mathbb{R}^D \to \mathbb{R}^{1\times N}$. For simplicity and efficiency, $\boldsymbol{B}^{(i)}$ and $\boldsymbol{C}^{(i)}$ are shared across all $1 \le i \le D_s$, and $\boldsymbol{A}^{(i)}$ is parameterized as a diagonal matrix.

4 Modular Sequence Modeling with SlotSSM

The SlotSSM proposed in Section 3 are designed to preserve modularity when the input is already separated into slots. In this section, we complement SlotSSM with a slot encoder that extracts slot representations from unstructured inputs (Section 4.1), and a slot mixer that introduces sparse interactions across slots (Section 4.2). We then present a sequence modeling architecture (Section 4.3) that encourages discovery of underlying modular processes by stacking these components.

4.1 Slot Encoder

We assume the unstructured input $\boldsymbol{x}_t$ at each time step t is represented as a sequence of M tokens:

$$\boldsymbol{x}_t = (\boldsymbol{x}_t^1, \ldots, \boldsymbol{x}_t^M), \quad \boldsymbol{x}_t^m \in \mathbb{R}^{D_x}. \tag{6}$$

For example, image inputs can be CNN feature maps (M is the number of cells in the feature map), or as embeddings of non-overlapping image patches (M is the number of patches), as proposed in ViT [12]. To extract K slot representations from $\boldsymbol{x}_t$, we use K learnable CLS [2] tokens $\{\mathrm{CLS}_t^k \in \mathbb{R}^{D_x}\}_{k=1}^K$ as queries and perform cross-attention with the input tokens through a Transformer [63]:

$$\{\mathrm{CLS}_t^k\}_{k=1}^K \leftarrow \mathrm{Transformer}\Big(\mathtt{q} = \{\mathrm{CLS}_t^k\}_{k=1}^K,\ \mathtt{kv} = \{\boldsymbol{x}_t^m\}_{m=1}^M\Big). \tag{7}$$

The Transformer also includes self-attention within the CLS tokens, allowing them to communicate with each other and capture information from different parts of the input, thereby facilitating the emergence of modularity. The slot representations are then obtained by applying a linear projection to the corresponding output embeddings of the CLS tokens:

$$\boldsymbol{s}_t^k = \mathrm{Linear}(\mathrm{CLS}_t^k), \quad k = 1, \ldots, K. \tag{8}$$

4.2 Slot Mixer

The slot encoder obtains slot decomposition purely based on single time steps, which can be suboptimal. In addition, the SlotSSM processes each slot fully independently, making it hard to correct mistakenly decomposed slots or model interarctions across slots. To resolve both issues, we interleave SlotSSM with slot mixers.

[2] Following the tradition of ViT, we use "CLS" to distingish learnable tokens from observation tokens.

The slot mixer consists of two residual blocks, and is applied to the outputs $\{\boldsymbol{y}_t^k\}_{k=1}^K$ of the SlotSSM. The first block introduces interaction across slots through self-attention [63], whereas the second block uses MLP to further process the gathered information within each slot:

$$\big(\boldsymbol{y}_t^1,\, \ldots,\, \boldsymbol{y}_t^K\big) \leftarrow \big(\boldsymbol{y}_t^1,\, \ldots,\, \boldsymbol{y}_t^K\big) + \mathrm{SelfAttn}\big(\mathrm{LN}(\boldsymbol{y}_t^1),\, \ldots,\, \mathrm{LN}(\boldsymbol{y}_t^K)\big)\ , \tag{9}$$

$$\big(\boldsymbol{y}_t^1,\, \ldots,\, \boldsymbol{y}_t^K\big) \leftarrow \big(\boldsymbol{y}_t^1,\, \ldots,\, \boldsymbol{y}_t^K\big) + \big(\mathrm{MLP}(\mathrm{LN}(\boldsymbol{y}_t^1)),\, \ldots,\, \mathrm{MLP}(\mathrm{LN}(\boldsymbol{y}_t^K))\big)\ . \tag{10}$$

Here, $\mathrm{LN}(\cdot)$ denotes layer normalization [2]. Because $\boldsymbol{y}_t^k$ carries information from the entire history of each slot, it provides the opportunity to refine the slot representations based on temporal dynamics.

4.3 Sequence Modeling Architecture

We now present a generic architecture for modeling sequences with modular underlying processes. Given a sequence of unstructured inputs $\boldsymbol{x}_{1:T}$, our goal is to obtain a set of K_l modular representations at each time step t and at each layer l that summarizes all underlying processes up to time t.

In general, the number of slots K_l at each layer can be different, potentially allowing fewer but more abstract slots at higher layers. To accommodate this, we insert a slot encoder wherever the number of slots changes, and repurpose it to extract a different number of slots from existing slot representations. This is achieved by treating the slots output from the previous layer as keys and values in Equation 7. When the number of slots does not change, we can simply copy the slots from the previous layer.

As shown in Figure 3, our proposed architecture stacks the (optional) slot encoder, SlotSSM, and slot mixer together at each layer. Variables at layer l are denoted with the subscript '$|l$'. The slot mixer's output from layer $l-1$, $\{\boldsymbol{x}_{t|l-1}^k\}_{k=1}^{K_{l-1}}$, serves as input to layer l. The initial input is $\{\boldsymbol{x}_{t|0}^k\}_{k=1}^{K_0}$, where $K_0 \coloneqq M$. The computations at each layer $l = 1, \ldots, L$ are:

$$\{\boldsymbol{s}_{t|l}^k\}_{k=1}^{K_l} = \mathrm{SlotEncoder}\Big(\{\boldsymbol{x}_{t|l-1}^k\}_{k=1}^{K_{l-1}}\Big)\ , \tag{11}$$

$$\{\boldsymbol{y}_{t|l}^k\}_{k=1}^{K_l},\ \{\boldsymbol{h}_{t|l}^k\}_{k=1}^{K_l} = \mathrm{SlotSSM}\Big(\{\boldsymbol{s}_{t|l}^k\}_{k=1}^{K_l},\ \{\boldsymbol{h}_{t-1|l}^k\}_{k=1}^{K_l}\Big)\ , \tag{12}$$

$$\{\boldsymbol{x}_{t|l}^k\}_{k=1}^{K_l} = \mathrm{SlotMixer}\Big(\{\boldsymbol{y}_{t|l}^k\}_{k=1}^{K_l}\Big)\ . \tag{13}$$

The final output $\{\boldsymbol{x}_{t|L}^k\}_{k=1}^{K_L}$ can be used for various tasks, such as predicting the next observation and the properties of underlying processes (*e.g.*, position, velocity).

5 Object-Centric Learning with SlotSSM

In this section, we present a concrete example of adapting the generic sequence modeling architecture proposed in Section 4 to solve a specific task. We consider the task of object-centric representation learning from unannotated videos of interacting objects, a typical example of sequences with modular underlying structures. The goal is to obtain a representation for each individual object that captures relevant attributes such as object position, size, shape, color, *etc*. without any object-level annotation.

5.1 Object-Centric SlotSSMs (OC-SlotSSMs)

Inspired by previous works [46, 67], we make slight modifications to our sequence modeling architecture to facilitate the discovery of modular structures. We call the resulting model OC-SlotSSMs. First, we use the same number of slots across all layers. It is thus unnecessary to have a slot encoder per layer. However, we find it helpful to still have it, but in another form that encourages iterative refinement of the slots. Specifically, we use the slots output from the previous layer $\{\boldsymbol{x}_{t|l-1}^k\}_{k=1}^K$ as queries, and provide the input tokens $\{\boldsymbol{x}_{t|0}^m\}_{m=1}^M$ as keys and values. Second, we introduce competition among slots in the attention layers of the slot encoder. We achieve this by using inverted attention [61, 67], which is essentially cross attention with the Softmax operation performed over the queries instead of the keys. This has the effect of softly assigning each input token to a slot, thereby promoting modularity. The computation at each layer $l = 1, \ldots, L$ can be summarized as follows:

$$\{\boldsymbol{s}^k_{t|l}\}^K_{k=1} = \mathrm{InvAttn}\Big(\mathtt{q} = \{\boldsymbol{x}^k_{t|l-1}\}^K_{k=1},\ \mathtt{kv} = \{\boldsymbol{x}^m_{t|0}\}^M_{m=1}\Big)\ , \tag{14}$$

$$\{\boldsymbol{y}^k_{t|l}\}^K_{k=1},\ \{\boldsymbol{h}^k_{t|l}\}^K_{k=1} = \mathrm{SlotSSM}\Big(\{\boldsymbol{s}^k_{t|l}\}^K_{k=1},\ \{\boldsymbol{h}^k_{t-1|l}\}^K_{k=1}\Big)\ , \tag{15}$$

$$\{\boldsymbol{x}^k_{t|l}\}^K_{k=1} = \mathrm{SlotMixer}\Big(\{\boldsymbol{y}^k_{t|l}\}^K_{k=1}\Big)\ . \tag{16}$$

We note that the queries in the first inverted attention layer are the learnable CLS tokens $\{\mathrm{CLS}^k_{t|0}\}^K_{k=1}$.

5.2 Training Pipeline

Following previous works in object-centric learning [46, 39, 57], we adopt an auto-encoding training pipeline. Given a sequence of video frames $\{\boldsymbol{o}_t \in \mathbb{R}^{H\times W\times 3}\}^T_{t=1}$, we obtain the input $\boldsymbol{x}_{t|0}$ to our sequence modeling architecture by applying a CNN encoder to each frame $\boldsymbol{o}_t$ and adding a positional embedding for each feature map cell. The output slots $\{\boldsymbol{x}^k_{t|L}\}^K_{k=1}$ are each decoded into an object image $\hat{\boldsymbol{o}}^k_t \in \mathbb{R}^{H\times W\times 3}$ and an alpha mask $\boldsymbol{\alpha}^k_t \in \mathbb{R}^{H\times W\times 1}$ by a spatial broadcast decoder [66]. The final reconstruction $\hat{\boldsymbol{o}}_t \in \mathbb{R}^{H\times W\times 3}$ is given by the alpha-composition of the object images:

$$\hat{\boldsymbol{o}}^k_t, \boldsymbol{\alpha}^k_t = \mathrm{Decoder}(\boldsymbol{x}^k_{t|L})\ , \quad \hat{\boldsymbol{o}}_t = \sum_{k=1}^K \frac{\exp(\boldsymbol{\alpha}^k_t)}{\sum_{j=1}^K \exp(\boldsymbol{\alpha}^j_t)} \cdot \hat{\boldsymbol{o}}^k_t\ . \tag{17}$$

The training objective is to minimize the reconstruction error $\mathcal{L} = \frac{1}{T}\sum_{t=1}^T \|\hat{\boldsymbol{o}}_t - \boldsymbol{o}_t\|_2^2$.

6 Related Work

State Space Models (SSMs). Popularized by S4 [27], SSMs have attracted growing interest in language modeling and as a sequence modeling framework in general. The original S4 follows the HiPPO theory [25] to parameterize and initialize the state transition matrices, which is quite mathematically involved. Most recent works have proposed simplified versions that use diagonal transition matrices [29, 26, 58] and pure RNN formulation (*i.e.*, without reliance on ODE discretization) [30, 50, 9]. Several works have proposed hybrid architectures of SSMs and Transformers to incorporate their complementary strengths [75, 48, 17, 32, 24]. In addition to language modeling, SSMs have been applied to various domains, including time-series generation [73], audio generation [19], visual classificiation and generation [49, 40, 32, 65, 74, 71], and reinforcement learning [8, 47, 10, 52]. Our study introduces the first SSM with inductive biases for modeling inherently modular processes.

Object-Centric Learning. Object-centric learning seeks to discover modular structures and independent mechanisms [20] such as objects and their relations from multi-object images and videos with weak or no supervision [3, 22, 38, 23, 15, 14, 16, 7, 45, 35, 41, 37, 6, 44, 11, 1, 64, 68, 53, 34]. Recent works are predominantly based on the Slot Attention [46] model, which uses a GRU [5] and competitive attention mechanisms to iteratively refine slot representations [54, 57, 39, 13, 69, 55, 36, 70]. However, GRUs and RNNs in general are prone to vanishing gradient issues [51], and the training must be done in a sequential way. These weaknesses render them incapable of scaling up to long-range videos. Additionally, hardware parallelization for video object-centric learning has been explored in [56], however, it incurs quadratic cost unlike ours. Our SlotSSMs framework can be specialized to address object-centric learning tasks effectively. By integrating SSMs at its core, SlotSSMs benefit from parallelizable training and possess remarkable long-term memory capabilities. Moreover, as a versatile framework, SlotSSMs are well-suited to tackle other tasks such as long-range visual reasoning.

7 Experiments

We present an extensive evaluation of our models across a variety of tasks. Section 7.1 illustrates the need for modular latent states through a multi-object video prediction task. Section 7.2 demonstrates the advantages of SlotSSMs over Transformers and RNNs using a newly proposed long-context reasoning benchmark. Section 7.3 investigates the object-centric learning capabilities of OC-SlotSSMs. Finally, Section 7.4 showcases the 3D visual reasoning capabilities using the CATER benchmark [18].

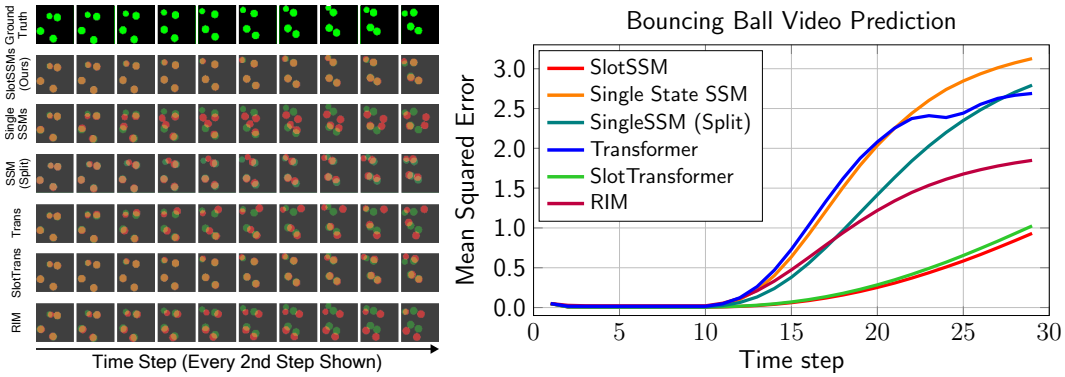


Figure 4: Multi-Object Video Prediction Task. *Left*: Generated video frames at every second step, showing 10 of the 20 total frames generated. Green color indicates ground-truth and red color indicates predictions. *Right*: MSE over a 20-frame autoregressive rollout, given 10 context frames. SlotSSM demonstrates its efficiency in modeling multi-object dynamics.

7.1 Multi-Object Video Prediction

We begin with a multi-object video modeling task to demonstrate the benefit of incorporating modularity into state space.

Dataset and Task. We utilize the bouncing balls video dataset introduced by [62], which consists of videos of white balls bouncing off each other in an empty window. Each ball has random initial positions, velocities, and masses, governing their interactions. The task is conditional video generation, specifically $p(\mathbf{x}_{T+1:T+W}|\mathbf{x}_{1:T})$. This task is inherently modular as it requires models to remember each object's attributes and interaction rules.

Experimental Setup. We train models on 20-frame sequences using teacher-forcing and binary cross-entropy loss. At test time, given $T = 10$ context frames, the model autoregressively predicts $W = 20$ future frames using its own outputs. Performance is evaluated using Mean Squared Error (MSE) between predicted and ground-truth images.

Models. We employ the SlotSSM architecture described in Section 4.3 We use the same number of slots across layers and apply the Slot Encoder only at the first layer. We compare our model against several baselines: Single State SSM, which shares the same architecture but uses a monolithic state; Single State SSM (Split), which uses a Single State SSM with multi-slot encoder and decoder—slots are concatenated in SSM, then split into multiple slots for the decoder; RIM[20], a slot-based RNN model with separate RNN weights per slot that introduces sparse slot updates and interactions based on input attention values; Transformer, a vanilla Transformer model with a single input embedding per time step; and SlotTransformer, a Transformer model with multiple input slots at each time step.

All models share the same encoder and decoder architectures. The encoder is a Transformer described in Section 4.1, using a single CLS token for single-state models. The decoder consists of three Transformer layers with self-attention for image patches and cross-attention to query the slots. We use six slots for all slot-based models. For RIM, we set $k = 4$ for top-k active modules as in the original paper. We carefully match hyperparameters across baselines to ensure comparable model sizes, except for RIM, which inherently requires a larger model due to separate RNN weights per slot. Additional implementation details are in Appendix C.

Results. Figure 4 compares model performances, showing that SlotSSM outperforms all baselines, including a slight improvement over SlotTransformer. The significant gap between SlotSSM and Single State SSM underscores the importance of modular slot states for effective multi-object dynamics learning, as also evidenced by the comparison between Transformer and SlotTransformer. SlotSSM also significantly outperforms Single State SSM (Split), which uses the same modular encoder and decoder, highlighting that modularity in temporal modeling—the core contribution of SlotSSM—is critical for improved performance. While the RIM model performs better than other single-state baselines, it still lags behind SlotSSM and SlotTransformer.

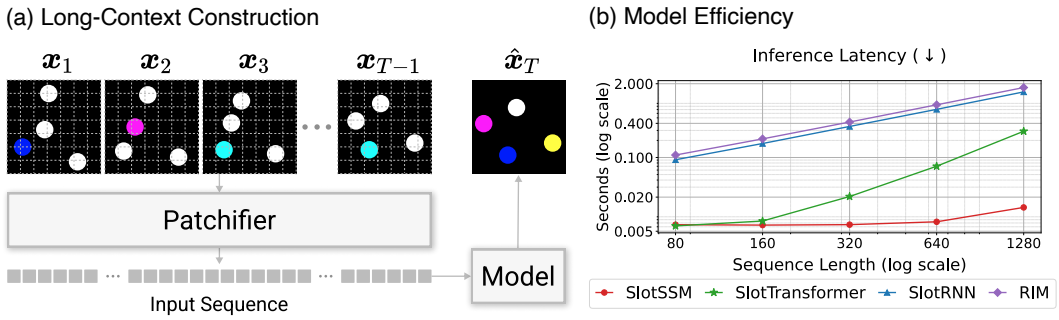


Figure 5: Long-Context Construction and Model Efficiency in the Blinking Color Balls Benchmark. *Left*: We construct long-sequence inputs by patchifying the context images. *Right*: Comparison of model inference latency with batch size 6. SlotSSM demonstrates computational efficiency for long-sequence processing tasks.

7.2 Long-Context Reasoning

We now evaluate the long-context reasoning capabilities of SlotSSM. To enable a rigorous assessment in a multi-object setting, we propose the novel Blinking Color Balls Benchmark.

Blinking Color Balls Benchmark. This benchmark has two variants—Earliest Color and Most Frequent Color—each consisting of image sequences with context images $\mathbf{x}_{1:T-1}$ and a target image $\mathbf{x}_T$. In each context image, one ball is randomly selected and assigned a non-white color from five options while others remain white. The coloring of balls in the target image $\mathbf{x}_T$ depends on the variant: in **Earliest Color**, each ball's color is its earliest assigned non-white color in the context (remaining white if none); in **Most Frequent Color**, each ball's color is the non-white color assigned most frequently during the context (ties broken by earliest assignment; remaining white if none).

To create a long-range reasoning task, we further patchify each context image into non-overlapping $P \times P$ patches and flatten them into a sequence of length P^2, as shown in Figure 5(a). With context length $T-1$, the total input sequence length is $L = (T-1) \times P^2$. Models must identify and track objects from partial patch views while remembering and counting color assignments, making the task highly challenging. The final task is to predict the target image given this long sequential input.

Experimental Setup. We evaluate models on Earliest Color with $T = 6$ and Most Frequent Color with $T \in \{6, 11\}$, using patch sizes $P \in \{4, 8, 16\}$. This yields input sequence lengths $L \in \{80, 160, 320, 640, 1280, 2560\}$. The Most Frequent Color with $T = 11$ requires stronger memorization and reasoning capabilities due to longer context and more color assignments, .

Models. We employ the same encoder, decoder, and the SlotSSM architectures as in Section 7.1. For slot encoding, each image patch is treated as an image and processed by the slot encoder. The slots from the last time step are provided to the decoder to predict the full target image. We compare our SlotSSM against several baselines: Single State SSM, SlotTransformer, and RIM. Additionally, we introduce a novel slot-based design called SlotRNNs, which shares RNN weights across slots and uses self-attention layers between time steps as the slot mixer. SlotRNNs can be viewed as a special case of RIMs with shared weights and dense state updates. Empirically, SlotRNNs exhibit more stable training and improved performance compared to RIMs. For fair comparison, all slot-based models use six slots, and we carefully match model sizes as in Section 7.1.

Results. Figure 6 shows that SlotSSM outperforms Single State SSM, SlotRNN, and RIM across all sequence lengths. For shorter sequences (80 and 160), Single State SSM and SlotRNN have relatively low error rates but degrade significantly beyond 320 frames. Surprisingly, RIM fails to generalize at any sequence length, likely due to optimization issues from separate weights per slot; our SlotRNN addresses this by sharing weights across slots while maintaining modularity. SlotTransformer performs competitively up to 640 frames. However, SlotSSM demonstrates superior long-range reasoning, especially at 1280 and 2560 frames, where other models cannot run due to memory or optimization constraints. Figure 5(b) highlights SlotSSM's computational efficiency. SlotTransformer's inference latency increases rapidly with sequence length due to quadratic complexity, SlotSSM maintains stable and efficient inference across all lengths. Due to SlotTransformer's high memory usage, we used a batch size of 6 for latency evaluation. Qualitative comparisons in Appendix B.3 provide further insights into the models' strengths and weaknesses.

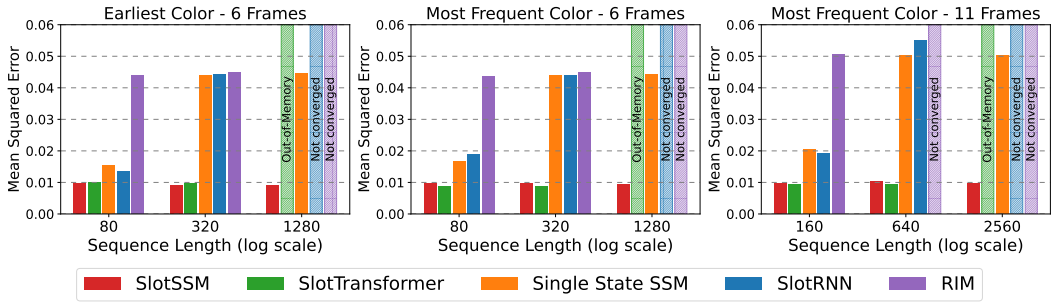


Figure 6: Long-Context Reasoning in Blinking Balls Benchmark. SlotSSM maintains consistent performance across sequence lengths from 80 to 2560, whereas baseline models show degraded performance or fail to complete training due to high memory and computational requirements.

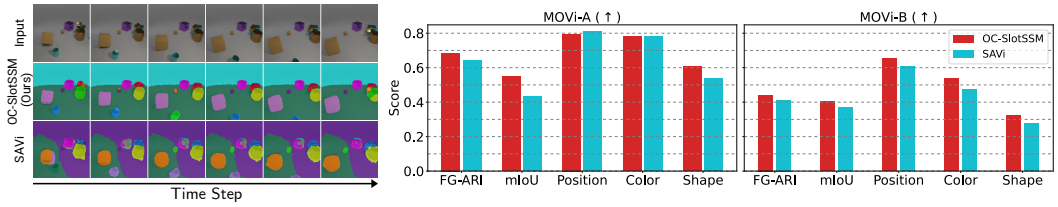


Figure 7: Object-Centric Learning Results. *Left*: Qualitative comparison of segmentation masks on MOVi-A. OC-SlotSSM demonstrate less object spliting and better boundary adherence. *Right*: Quantitative evaluation on unsupervised object segmentation and attribute prediction. OC-SlotSSM outperforms SAVi on most metrics.

7.3 Unsupervised Object-Centric Learning

In this section, we evaluate the performance of the Object-Centric SlotSSMs (OC-SlotSSM) variant in unsupervised object-centric representation learning.

Datasets. We evaluate OC-SlotSSM on the MOVi-A and MOVi-B subsets of the MOVi video dataset [21], which contain videos of up to 10 objects moving in a 3D environment. MOVi-B adds complexity over MOVi-A by including a wider variety of object types and multi-colored backgrounds.

Tasks. Following prior object-centric learning works [46, 36], we evaluate models on two downstream tasks: unsupervised object segmentation and attribute prediction. For segmentation, we report FG-ARI and mIoU metrics. For attribute prediction, we measure the quality of representations by inferring object properties: we report prediction accuracy for discrete attributes (e.g., object shape) and R^2 for continuous attributes (e.g., object position).

Models. We compare OC-SlotSSM to SAVi [39], an RNN-based object-centric learning approach. Both models use a CNN encoder to extract image features as input tokens $\boldsymbol{x}t \mid 0^m m = 1^M$, which are processed by their respective attention mechanisms—inverted attention in OC-SlotSSM and slot attention in SAVi—to produce slots. These slots are then used to reconstruct the image and generate per-object segmentation masks via a spatial broadcast decoder, with reconstruction as the training objective. For unsupervised object segmentation, we directly use the object masks obtained during training. For attribute prediction, we match slots to object IDs using Hungarian matching based on segmentation masks, then use linear heads and 2-layer MLPs to predict discrete and continuous attributes, respectively, keeping the slots frozen.

Results. Results in Figure 7 demonstrate that OC-SlotSSM consistently outperforms SAVi in unsupervised object segmentation on both MOVi-A and MOVi-B. The qualitative comparison (Figure 7, left) shows that OC-SlotSSM generates masks with tighter object boundaries and fewer object splitting, which also leads to improved attribute prediction accuracy (Figure 7, right). Furthermore, we empirically found that OC-SlotSSM exhibits superior stability during training compared to SAVi, which tends to collapse into a single slot representing the entire scene when trained long enough. This collapse is not reflected in the validation loss, so we apply early stopping based on manual inspection. In contrast, OC-SlotSSM does not suffer from this instability, demonstrating its robustness in learning object-centric representations.

Table 1: Performance on CATER Snitch Localization Task.

Model	No Pre-train		Pre-train	
	Top-1 Acc (%)	Top-5 Acc (%)	Top-1 Acc (%)	Top-5 Acc (%)
Single State SSM	10.27	27.21	41.15	65.70
SlotTransformer	41.09	62.24	49.21	70.24
SlotSSM	25.64	45.03	54.73	74.42
OC-SlotSSM	**61.58**	**84.00**	**69.27**	**90.48**

7.4 3D Visual Reasoning

Finally, we explore the application of SlotSSM and OC-SlotSSM to 3D visual reasoning tasks using the CATER benchmark [18].

CATER Benchmark. CATER consists of 300-frame video episodes of objects moving in a 3D environment. The movement can lead to partial occlusions and even complete coverage of smaller objects by larger ones. The primary task is snitch localization—predicting the golden snitch's location in the final frame. The snitch is always present but may be occluded. Models must reason about its location based on the last visible position and other objects' movements. Success in this task demonstrates models' capacity for complex visual reasoning in dynamic 3D environments.

Experimental Setup. We consider two experiment settings: direct training and pre-training + fine-tuning. In direct training, models are trained end-to-end on the snitch localization task. In pre-training + fine-tuning, models are first pre-trained on video inputs using a reconstruction objective, then fine-tuned on the task-specific signal. During pre-training, we randomly sample 32 frames from the 300-frame videos. For direct training and fine-tuning, we split the sequence into 50 non-overlapping segments of 6 frames each, randomly selecting one frame from each to create a 50-frame sequence spanning the entire video. At test time, we evenly sample 50 frames by skipping every 6 frames. The snitch's final location is quantized into a 6×6 grid, framing the problem as a classification task.

Models. We evaluate the performance of SlotSSM, OC-SlotSSM, Single State SSM, and SlotTransformer. We exclude RNN-based baselines, as our preliminary experiments reveal that they are unstable when handling long video inputs and prone to collapse to a constant output. For the visual pre-training setting, we employ a spatial broadcast decoder to reconstruct the input images. During downstream training/fine-tuning, we feed the slots from the final step to a transformer predictor with single CLS token, followed by a linear layer on the output CLS token to predict the snitch's position.

Results. Table 1 presents the Top-1 and Top-5 accuracy on the CATER Snitch Localization task. Consistent with our previous findings, SlotSSM outperforms Single State SSM, highlighting the importance of modular latent structures. Comparing SlotSSM with SlotTransformer, we see notable differences between direct training and pre-training settings: in direct training, SlotTransformer surpasses SlotSSM, possibly due to optimization advantages from direct access to all previous states; however, SlotSSM benefits more from pre-training, likely due to the explicit memory capacity of SSM states, consequently, pre-trained SlotSSMs outperforming their SlotTransformer counterparts.

Remarkably, OC-SlotSSM achieves the highest accuracy, outperforming all baselines by a large margin in both direct training and pre-training settings. This performance gain may be attributed to the explicit decomposition into object-centric representations, which facilitates reasoning about object properties, relationships, and interactions.

8 Conclusion

In this work, we presented SlotSSMs a novel approach to incorporating modular structure and inductive biases into State Space Models for improved sequence modeling. By maintaining a collection of independent slot vectors and performing state transitions independently per slot with sparse interactions via self-attention, SlotSSMs effectively captures the inherent modularity present in many real-world processes. The experimental results in object-centric video understanding and video prediction tasks demonstrate the substantial performance gains offered by SlotSSMs over existing sequence modeling methods.

Acknowledgements

This research was supported by GRDC (Global Research Development Center) Cooperative Hub Program (RS-2024-00436165) and Brain Pool Plus Program (No. 2021H1D3A2A03103645) through the National Research Foundation of Korea (NRF) funded by the Ministry of Science and ICT.

References

[1] Titas Anciukevicius, Christoph H Lampert, and Paul Henderson. Object-centric image generation with factored depths, locations, and appearances. *arXiv preprint arXiv:2004.00642*, 2020.

[2] Jimmy Ba, Jamie Ryan Kiros, and Geoffrey Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[3] Christopher P Burgess, Loic Matthey, Nicholas Watters, Rishabh Kabra, Irina Higgins, Matt Botvinick, and Alexander Lerchner. Monet: Unsupervised scene decomposition and representation. *arXiv preprint arXiv:1901.11390*, 2019.

[4] Chang Chen, Yi-Fu Wu, Jaesik Yoon, and Sungjin Ahn. TransDreamer: Reinforcement learning with Transformer world models. In *Deep RL Workshop NeurIPS 2021*, 2021.

[5] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

[6] Eric Crawford and Joelle Pineau. Exploiting spatial invariance for scalable unsupervised object tracking. *arXiv preprint arXiv:1911.09033*, 2019.

[7] Eric Crawford and Joelle Pineau. Spatially invariant unsupervised object detection with convolutional neural networks. In *Proceedings of AAAI*, 2019.

[8] Shmuel Bar David, Itamar Zimerman, Eliya Nachmani, and Lior Wolf. Decision S4: Efficient sequence-based RL via state spaces layers. In *International Conference on Learning Representations*, 2023.

[9] Soham De, Samuel L Smith, Anushan Fernando, Aleksandar Botev, George Cristian-Muraru, Albert Gu, Ruba Haroun, Leonard Berrada, Yutian Chen, Srivatsan Srinivasan, et al. Griffin: Mixing gated linear recurrences with local attention for efficient language models. *arXiv preprint arXiv:2402.19427*, 2024.

[10] Fei Deng, Junyeong Park, and Sungjin Ahn. Facing off world model backbones: RNNs, Transformers, and S4. *Advances in Neural Information Processing Systems*, 36, 2024.

[11] Fei Deng, Zhuo Zhi, Donghun Lee, and Sungjin Ahn. Generative scene graph networks. In *ICLR 2021: The Ninth International Conference on Learning Representations*, 2021.

[12] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021.

[13] Gamaleldin F. Elsayed, Aravindh Mahendran, Sjoerd van Steenkiste, Klaus Greff, Michael Curtis Mozer, and Thomas Kipf. Savi++: Towards end-to-end object-centric learning from real-world videos. *ArXiv*, abs/2206.07764, 2022.

[14] Martin Engelcke, Oiwi Parker Jones, and Ingmar Posner. GENESIS-V2: Inferring unordered object representations without iterative refinement. *arXiv preprint arXiv:2104.09958*, 2021.

[15] Martin Engelcke, Adam R. Kosiorek, Oiwi Parker Jones, and Ingmar Posner. GENESIS: Generative scene inference and sampling with object-centric latent representations. In *ICLR 2020 : Eighth International Conference on Learning Representations*, 2020.

[16] SM Ali Eslami, Nicolas Heess, Theophane Weber, Yuval Tassa, David Szepesvari, and Geoffrey E Hinton. Attend, infer, repeat: Fast scene understanding with generative models. In *Advances in Neural Information Processing Systems*, pages 3225–3233, 2016.

[17] Daniel Y Fu, Tri Dao, Khaled Kamal Saab, Armin W Thomas, Atri Rudra, and Christopher Ré. Hungry Hungry Hippos: Towards language modeling with state space models. In *International Conference on Learning Representations*, 2023.

[18] Rohit Girdhar and Deva Ramanan. CATER: A diagnostic dataset for Compositional Actions and TEmporal Reasoning. In *International Conference on Learning Representations*, 2020.

[19] Karan Goel, Albert Gu, Chris Donahue, and Christopher Ré. It's raw! Audio generation with state-space models. In *International Conference on Machine Learning*, 2022.

[20] Anirudh Goyal, Alex Lamb, Jordan Hoffmann, Shagun Sodhani, Sergey Levine, Yoshua Bengio, and Bernhard Schölkopf. Recurrent independent mechanisms. *ArXiv*, abs/1909.10893, 2021.

[21] Klaus Greff, Francois Belletti, Lucas Beyer, Carl Doersch, Yilun Du, Daniel Duckworth, David Fleet, Dan Gnanapragasam, Florian Golemo, Charles Herrmann, Thomas Kipf, Abhijit Kundu, Dmitry Lagun, Issam H. Laradji, Hsueh-Ti Liu, Henning Meyer, Yishu Miao, Derek Nowrouzezahrai, Cengiz Oztireli, Etienne Pot, Noha Radwan, Daniel Rebain, Sara Sabour, Mehdi S. M. Sajjadi, Matan Sela, Vincent Sitzmann, Austin Stone, Deqing Sun, Suhani Vora, Ziyu Wang, Tianhao Wu, Kwang Moo Yi, Fangcheng Zhong, and Andrea Tagliasacchi. Kubric: A scalable dataset generator. *arXiv preprint arXiv:2203.03570*, 2022.

[22] Klaus Greff, Raphaël Lopez Kaufmann, Rishab Kabra, Nick Watters, Chris Burgess, Daniel Zoran, Loic Matthey, Matthew Botvinick, and Alexander Lerchner. Multi-object representation learning with iterative variational inference. *arXiv preprint arXiv:1903.00450*, 2019.

[23] Klaus Greff, Sjoerd van Steenkiste, and Jürgen Schmidhuber. Neural expectation maximization. In *Advances in Neural Information Processing Systems*, pages 6691–6701, 2017.

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

[25] Albert Gu, Tri Dao, Stefano Ermon, Atri Rudra, and Christopher Ré. HiPPO: Recurrent memory with optimal polynomial projections. In *Advances in Neural Information Processing Systems*, 2020.

[26] Albert Gu, Karan Goel, Ankit Gupta, and Christopher Ré. On the parameterization and initialization of diagonal state space models. In *Advances in Neural Information Processing Systems*, 2022.

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

[28] Albert Gu, Isys Johnson, Karan Goel, Khaled Kamal Saab, Tri Dao, Atri Rudra, and Christopher Ré. Combining recurrent, convolutional, and continuous-time models with linear state space layers. In *Advances in Neural Information Processing Systems*, 2021.

[29] Ankit Gupta, Albert Gu, and Jonathan Berant. Diagonal state spaces are as effective as structured state spaces. In *Advances in Neural Information Processing Systems*, 2022.

[30] Ankit Gupta, Harsh Mehta, and Jonathan Berant. Simplifying and understanding state space models with diagonal linear RNNs. *arXiv preprint arXiv:2212.00768*, 2022.

[31] Li Hu. Animate anyone: Consistent and controllable image-to-video synthesis for character animation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8153–8163, 2024.

[32] Md Mohaiminul Islam and Gedas Bertasius. Long movie clip classification with state-space video models. In *ECCV*, 2022.

[33] Yasamin Jafarian and Hyun Soo Park. Learning high fidelity depths of dressed humans by watching social media dance videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12753–12762, 2021.

[34] Baoxiong Jia, Yu Liu, and Siyuan Huang. Improving Object-centric Learning with Query Optimization. In *International Conference on Learning Representations (ICLR)*, 2023.

[35] Jindong Jiang and Sungjin Ahn. Generative neurosymbolic machines. In *Advances in Neural Information Processing Systems*, 2020.

[36] Jindong Jiang, Fei Deng, Gautam Singh, and Sungjin Ahn. Object-centric slot diffusion. *Advances in Neural Information Processing Systems*, 36, 2024.

[37] Jindong Jiang, Sepehr Janghorbani, Gerard De Melo, and Sungjin Ahn. Scalor: Generative world models with scalable object representations. In *International Conference on Learning Representations*, 2019.

[38] Rishabh Kabra, Daniel Zoran, Goker Erdogan, Loic Matthey, Antonia Creswell, Matthew Botvinick, Alexander Lerchner, and Christopher P. Burgess. Simone: View-invariant, temporally-abstracted object representations via unsupervised video decomposition. *arXiv preprint arXiv:2106.03849*, 2021.

[39] Thomas Kipf, Gamaleldin F. Elsayed, Aravindh Mahendran, Austin Stone, Sara Sabour, Georg Heigold, Rico Jonschkowski, Alexey Dosovitskiy, and Klaus Greff. Conditional Object-Centric Learning from Video. *arXiv preprint arXiv:2111.12594*, 2021.

[40] David M Knigge, David W Romero, Albert Gu, Efstratios Gavves, Erik J Bekkers, Jakub Mikolaj Tomczak, Mark Hoogendoorn, and Jan-jakob Sonke. Modelling long range dependencies in ND: From task-specific to a general purpose CNN. In *International Conference on Learning Representations*, 2023.

[41] Adam Kosiorek, Hyunjik Kim, Yee Whye Teh, and Ingmar Posner. Sequential attend, infer, repeat: Generative modelling of moving objects. In *Advances in Neural Information Processing Systems*, pages 8606–8616, 2018.

[42] Yong Jae Lee, Joydeep Ghosh, and Kristen Grauman. Discovering important people and objects for egocentric video summarization. In *2012 IEEE conference on computer vision and pattern recognition*, pages 1346–1353. IEEE, 2012.

[43] Zhiqi Li, Wenhai Wang, Hongyang Li, Enze Xie, Chonghao Sima, Tong Lu, Yu Qiao, and Jifeng Dai. Bevformer: Learning bird's-eye-view representation from multi-camera images via spatiotemporal transformers. In *European conference on computer vision*, pages 1–18. Springer, 2022.

[44] Zhixuan Lin, Yi-Fu Wu, Skand Vishwanath Peri, Jindong Jiang, and Sungjin Ahn. Improving generative imagination in object-centric world models. In *International Conference on Machine Learning*, pages 4114–4124, 2020.

[45] Zhixuan Lin, Yi-Fu Wu, Skand Vishwanath Peri, Weihao Sun, Gautam Singh, Fei Deng, Jindong Jiang, and Sungjin Ahn. Space: Unsupervised object-oriented scene representation via spatial attention and decomposition. In *International Conference on Learning Representations*, 2020.

[46] Francesco Locatello, Dirk Weissenborn, Thomas Unterthiner, Aravindh Mahendran, Georg Heigold, Jakob Uszkoreit, Alexey Dosovitskiy, and Thomas Kipf. Object-centric learning with slot attention, 2020.

[47] Chris Lu, Yannick Schroecker, Albert Gu, Emilio Parisotto, Jakob Foerster, Satinder Singh, and Feryal Behbahani. Structured state space models for in-context reinforcement learning. *arXiv preprint arXiv:2303.03982*, 2023.

[48] Harsh Mehta, Ankit Gupta, Ashok Cutkosky, and Behnam Neyshabur. Long range language modeling via gated state spaces. In *International Conference on Learning Representations*, 2023.

[49] Eric Nguyen, Karan Goel, Albert Gu, Gordon Downs, Preey Shah, Tri Dao, Stephen Baccus, and Christopher Ré. S4ND: Modeling images and videos as multidimensional signals with state spaces. In *Advances in Neural Information Processing Systems*, 2022.

[50] Antonio Orvieto, Samuel L Smith, Albert Gu, Anushan Fernando, Caglar Gulcehre, Razvan Pascanu, and Soham De. Resurrecting recurrent neural networks for long sequences. In *International Conference on Machine Learning*, 2023.

[51] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. In *International Conference on Machine Learning*, 2013.

[52] Mohammad Reza Samsami, Artem Zholus, Janarthanan Rajendran, and Sarath Chandar. Mastering memory tasks with world models. In *The Twelfth International Conference on Learning Representations*, 2024.

[53] Maximilian Seitzer, Max Horn, Andrii Zadaianchuk, Dominik Zietlow, Tianjun Xiao, Carl-Johann Simon-Gabriel, Tong He, Zheng Zhang, Bernhard Scholkopf, Thomas Brox, and Francesco Locatello. Bridging the gap to real-world object-centric learning. *arXiv preprint arXiv:2209.14860*, 2022.

[54] Gautam Singh, Fei Deng, and Sungjin Ahn. Illiterate dall-e learns to compose. In *International Conference on Learning Representations*, 2022.

[55] Gautam Singh, Yeongbin Kim, and Sungjin Ahn. Neural Systematic Binder. In *International Conference on Learning Representations*, 2023.

[56] Gautam Singh, Yue Wang, Jiawei Yang, Boris Ivanovic, Sungjin Ahn, Marco Pavone, and Tong Che. Parallelized spatiotemporal binding, 2024.

[57] Gautam Singh, Yi-Fu Wu, and Sungjin Ahn. Simple unsupervised object-centric learning for complex and naturalistic videos. *arXiv preprint arXiv:2205.14065*, 2022.

[58] Jimmy T.H. Smith, Andrew Warrington, and Scott Linderman. Simplified state space layers for sequence modeling. In *International Conference on Learning Representations*, 2023.

[59] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, Vijay Vasudevan, Wei Han, Jiquan Ngiam, Hang Zhao, Aleksei Timofeev, Scott Ettinger, Maxim Krivokon, Amy Gao, Aditya Joshi, Yu Zhang, Jonathon Shlens, Zhifeng Chen, and Dragomir Anguelov. Scalability in perception for autonomous driving: Waymo open dataset. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[60] Yi Tay, Mostafa Dehghani, Samira Abnar, Yikang Shen, Dara Bahri, Philip Pham, Jinfeng Rao, Liu Yang, Sebastian Ruder, and Donald Metzler. Long Range Arena: A benchmark for efficient Transformers. In *International Conference on Learning Representations*, 2021.

[61] Yao-Hung Hubert Tsai, Nitish Srivastava, Hanlin Goh, and Ruslan Salakhutdinov. Capsules with inverted dot-product attention routing. In *International Conference on Learning Representations*, 2020.

[62] Sjoerd Van Steenkiste, Michael Chang, Klaus Greff, and Jürgen Schmidhuber. Relational neural expectation maximization: Unsupervised discovery of objects and their interactions. *arXiv preprint arXiv:1802.10353*, 2018.

[63] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, 2017.

[64] Julius von Kügelgen, Ivan Ustyuzhaninov, Peter V. Gehler, Matthias Bethge, and Bernhard Schölkopf. Towards causal generative scene models via competition of experts. *arXiv preprint arXiv:2004.12906*, 2020.

[65] Jue Wang, Wentao Zhu, Pichao Wang, Xiang Yu, Linda Liu, Mohamed Omar, and Raffay Hamid. Selective structured state-spaces for long-form video understanding. In *CVPR*, 2023.

[66] Nicholas Watters, Loic Matthey, Christopher P. Burgess, and Alexander Lerchner. Spatial broadcast decoder: A simple architecture for learning disentangled representations in vaes. *arXiv preprint arXiv:1901.07017*, 2019.

[67] Yi-Fu Wu, Klaus Greff, Gamaleldin Fathy Elsayed, Michael Curtis Mozer, Thomas Kipf, and Sjoerd van Steenkiste. Inverted-attention transformers can learn object representations: Insights from slot attention. In *UniReps: the First Workshop on Unifying Representations in Neural Models*, 2023.

[68] Yi-Fu Wu, Jaesik Yoon, and Sungjin Ahn. Generative video transformer: Can objects be the words? In *International Conference on Machine Learning*, pages 11307–11318. PMLR, 2021.

[69] Ziyi Wu, Nikita Dvornik, Klaus Greff, Thomas Kipf, and Animesh Garg. Slotformer: Unsupervised visual dynamics simulation with object-centric models. *arXiv preprint arXiv:2210.05861*, 2022.

[70] Ziyi Wu, Jingyu Hu, Wuyue Lu, Igor Gilitschenski, and Animesh Garg. Slotdiffusion: Object-centric generative modeling with diffusion models. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

[71] Jing Nathan Yan, Jiatao Gu, and Alexander M. Rush. Diffusion models without attention, 2023.

[72] Hanrong Ye, Haotian Zhang, Erik Daxberger, Lin Chen, Zongyu Lin, Yanghao Li, Bowen Zhang, Haoxuan You, Dan Xu, Zhe Gan, et al. Mm-ego: Towards building egocentric multimodal llms. *arXiv preprint arXiv:2410.07177*, 2024.

[73] Linqi Zhou, Michael Poli, Winnie Xu, Stefano Massaroli, and Stefano Ermon. Deep latent state space models for time-series generation. In *International Conference on Machine Learning*, 2023.

[74] Lianghui Zhu, Bencheng Liao, Qian Zhang, Xinlong Wang, Wenyu Liu, and Xinggang Wang. Vision mamba: Efficient visual representation learning with bidirectional state space model. *arXiv preprint arXiv:2401.09417*, 2024.

[75] Simiao Zuo, Xiaodong Liu, Jian Jiao, Denis Charles, Eren Manavoglu, Tuo Zhao, and Jianfeng Gao. Efficient long sequence modeling via state space augmented Transformer. *arXiv preprint arXiv:2212.08136*, 2022.

A Limitations & Broader Impact

Limitations SlotSSMs' success illustrates the importance of designing architectures that align with the problem domain's underlying modular structure. It also paves the way for future research in modular and object-centric sequence modeling. However, it has some limitations that future studies could address. First, compared to Transformer architectures, we find that SlotSSMs could benefit more from a pre-training phase in visual reasoning tasks. For example, in the 3D visual reasoning task, SlotSSMs underperform Transformer models when trained without pretraining. However, when combined with task-free pretraining, SlotSSMs demonstrate significant improvement, enabling them to outperform Transformer models. We note that this effect of task-free pre-training is more prominent in SlotSSMs than in Transformer baselines. This suggests that for tasks with sparse training signals, the sequential nature of SlotSSM performs better with a pre-training phase to learn to effectively utilize information from all time steps. We believe this phenomenon is worth further investigation in future research. Second, while the proposed architecture is applicable beyond video modeling—for example, to other modalities such as text and audio—this study has not explored these possibilities. It remains a matter for future work. Thrid, due to our academic research lab's computing resource constraints, we were unable to significantly scale up the proposed model to industry-scale in terms of model size and data size. Scaling up SlotSSMs could uncover additional properties or limitations that are not evident at the current scale of experimentation. Lastly, future studies should investigate the effect of increased visual complexity in videos. As an initial step, we present a preliminary exploration in Appendix D.2, where SlotSSMs are applied to natural video scenes. These experiments illustrate how modularity can emerge through the independent mechanisms of SlotSSMs in real-world scenarios. We hope these findings will inspire future research on the industry-scale adoption of SlotSSMs.

Impact Statement The introduction of SlotSSMs, a novel framework that incorporates independent mechanisms into State Space Models (SSMs), has the potential to significantly impact the field of sequence modeling. By leveraging the modular structure inherent in many real-world processes, SlotSSMs offers a more intuitive and effective approach to modeling long-range temporal dependencies in object-centric video understanding and prediction tasks. The substantial performance gains demonstrated by SlotSSMs over existing sequence modeling methods highlight the importance of designing architectures that align with the underlying structure of the problem domain. This breakthrough could lead to the development of more efficient and accurate models for a wide range of applications, such as robotics, autonomous vehicles, and video surveillance systems. Moreover, the success of SlotSSMs in capturing the modular nature of real-world processes could inspire further research into modular and object-centric sequence modeling. This could result in the development of even more advanced architectures that can better handle the complexity and diversity of real-world data. Because this is a general backbone architecture for sequence modeling, it doesn't raise direct ethical concerns. However, its ethical implications depend on the way downstream application developers use the model.

B Blinking Color Balls Benchmark

B.1 Motivation

Real-world videos are often inherently modular, involving multiple dynamic entities and their interactions across time. However, existing long-range reasoning tasks, such as those in the Long-Range Arena Benchmark [60], are typically designed to focus on single-object settings and recognizing a single dynamic pattern in the observations. To bridge this gap and facilitate more comprehensive evaluation, we propose the Blinking Color Balls Benchmark, a long-range visual reason benchmark desgined in a multi-object setting.

B.2 Dataset Design

We provide an illustrative example of the dataset design in Figure 8. Each episode of the dataset contains a context-target pair $(\mathbf{x}_{1:T-1}, \mathbf{x}_T)$. At each timestep in $\mathbf{x}_{1:T-1}$, all bouncing balls are first colored white, and then one ball is randomly picked and colored with one of 5 non-white colors. This process is repeated for all context frames, and it is represented in the rows in Figure 8(top). Note that the object picking and coloring are performed independently for each timestep, thus one ball could

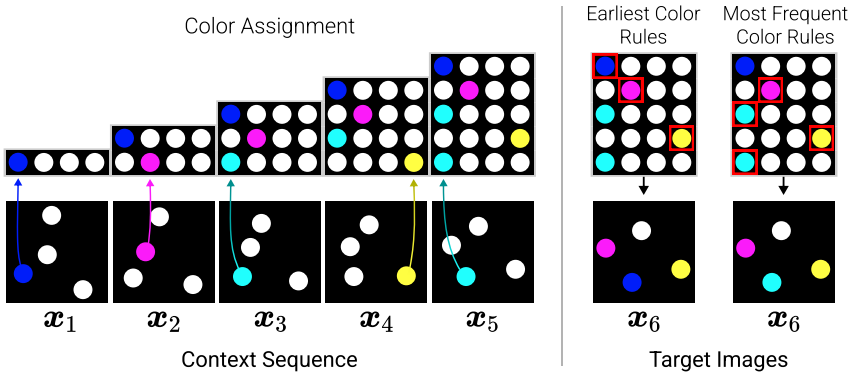


Figure 8: Blinking Color Balls Benchmark Overview. *Left*: Context frames with independent random ball picking and color assignments for each frame. Top figures indicate the sequential color assignment. *Right*: Target image for the Earliest Color and Most Frequent Color variants. Top figures indicate the color assignment rules.

be selected none or multiple times and colorized with the same or different colors across different timesteps.

The target images are then constructed with two rules: Earliest Color and Most Frequent Color. The Earliest Color rule picks the earliest non-white color assigned to the ball as the final color, while the Most Frequent Color rule counts the assignment of each non-white color and picks the color with the highest count (if there are ties, the earlier color among the highest is chosen). In Figure 8, we differentiate the two datasets using the same context sequence, which will result in different target images based on the rule. Note that regardless of the color assignment, the objects are moving and follow the physical bouncing rules throughout the full sequence. More image samples can be found in Figure 9.

Finally, as illustrated in Figure 5(a), we transform the conditional image generation task into a long-range reasoning task by using patchified context images as input. Instead of providing the $T-1$ context images directly to the model, we flatten non-overlapping patches of the original images to create a long input sequence. Given $P \times P$ patches per image, the context length becomes $L = (T-1) \times P^2$. Note that patchification is used intentionally to construct long sequences for the benchmark; SlotSSMs in general do not inherently require patchified inputs and instead use a Slot Encoder to extract slots as input at each time step.

B.3 Challenges and Qualitative Comparison

The Blinking Color Balls tasks pose significant challenges for the models, as they are required to reason about the object movement and color assignment rules from partial views of objects in temporally distant image patches. We can define two levels of challenges: (1) identifying the objects from image patches and predicting their future positions based on their dynamics, and (2) determining the final color assignment of each object based on the given rules. The first challenge is relatively simple, as it primarily involves learning the dynamics of objects from the past two frames prior to the target time step. However, the second challenge is particularly difficult, as it requires the model

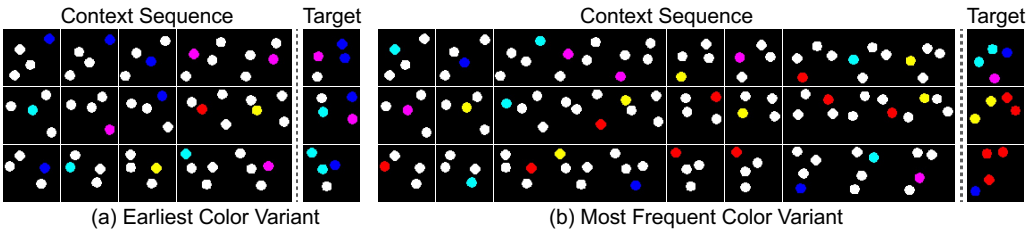


Figure 9: Blinking Color Balls Samples.

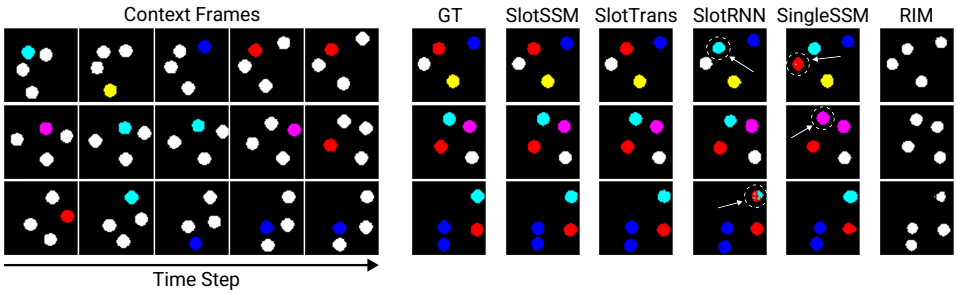


Figure 10: Blinking Color Balls Qualitative Comparison. Results shown for the Most Frequent Color variant with a sequence length of 80 frames.

to reason over the entire input sequence, necessitating the identification of an object's history from partially observed patches in a long-range context.

Figure 10 presents a qualitative comparison of the models' performance on the task. The results reveal a clear categorization of the models based on their capability to address the two levels of challenges. The baseline RIM model successfully predicts the object positions in the target image but struggles with learning the color assignment rules. Consequently, it predicts the color white that generally have the highest appearance probability for all objects. Note that the rendered images are based on the argmax of the logits over the color categories. Models such as SlotRNN and Single State SSM demonstrate the ability to learn color assignments, but they make mistakes in some cases. In contrast, SlotSSM and SlotTransformer successfully achieve both accurate position prediction and color assignment.

C Additional Implementation Details

C.1 SlotSSMs and OC-SlotSSMs

Slot Encoder. The main difference between the SlotSSMs and OC-SlotSSMs variants is in the design of the *Slot Encoders* as illustrated in Figure 11. The Slot Encoder in SlotSSMs is implemented as a multi-layer transformer with self-attention and cross-attention modules. Given the input tokens $\mathcal{X}_t = \{\boldsymbol{x}_t^m\}_{m=1}^M$, the structure of each layer in the Slot Encoder can be delineated into three modules:

$$\mathcal{C}_t = \text{SelfAttn}(\mathcal{C}_t)\,, \tag{18}$$

$$\mathcal{C}_t = \text{CrossAttn}\left(\texttt{q} = \mathcal{C}_t,\ \texttt{kv} = \mathcal{X}_t\right)\,, \tag{19}$$

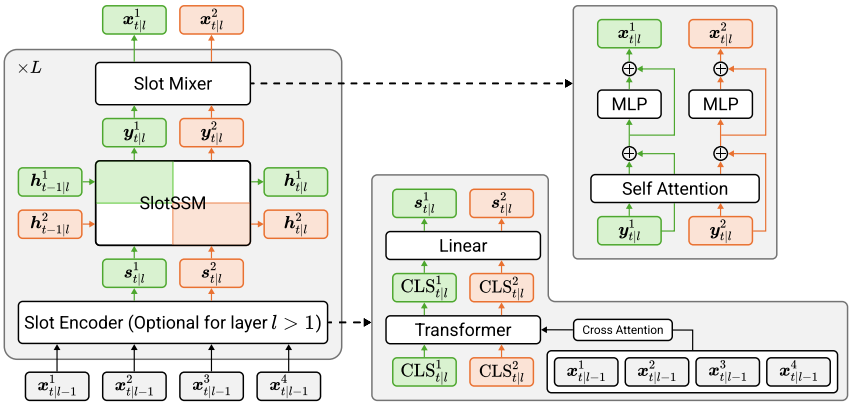
$$\mathcal{C}_t = \text{MLP}(\mathcal{C}_t)\,. \tag{20}$$

We use 3 layers in all our experiments. Note that we also apply skip connections and layer normalization in the input for all three modules, but have omitted them in the equations for brevity. The regular cross-attention used here employs softmax normalization over the attention weights applied to the input tokens:

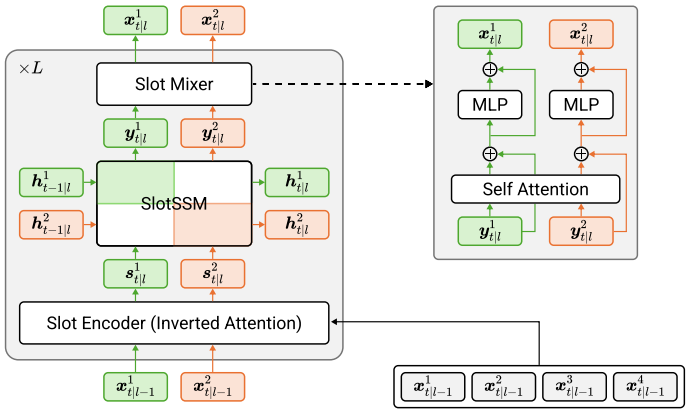
$$Q = W_Q(\mathcal{C}_t), \quad K = W_K(\mathcal{X}_t), \quad V = W_V(\mathcal{X}_t)\,, \tag{21}$$

$$\mathcal{C}_t^{\text{out}} = \texttt{softmax}\left(\frac{QK^T}{\sqrt{D}}, \quad \texttt{axis='keys'}\right) V\,. \tag{22}$$

In the OC-SlotSSMs layers, the *Slot Encoder* is implemented as a single inverted attention layer. This layer differs from the regular cross attention by the way attention weights are normalized:



(a) SlotSSMs



(b) OC-SlotSSMs

Figure 11: SlotSSMs vs OC-SlotSSMs.

$$Q = W_Q(\mathcal{C}_t), \quad K = W_K(\mathcal{X}_t), \quad V = W_V(\mathcal{X}_t)\,, \tag{23}$$

$$A = \texttt{softmax}\left(\frac{QK^T}{\sqrt{D}}, \quad \texttt{axis='queries'}\right)\,, \tag{24}$$

$$A_{i,j} = \frac{A_{i,j}}{\sum_{j=1}^{N_K} A_{i,j}}, \tag{25}$$

$$\mathcal{C}_t^{\text{out}} = AV\,. \tag{26}$$

The inverted attention layer applies softmax normalization over the queries, introducing a competition among the query tokens over the attention to the input tokens and thereby promoting disentanglement for the input tokens.

SSM Blocks. For the implementation of the SSM models, we leverage recent advances in linear state space models and design our SSM block in SlotSSM based on the Mamba architecture [24]. The block-diagonal transition of slots is implemented as parallel runs of SSM blocks that share the same model weights.

$$\{\boldsymbol{y}_{t|l}^k\}_{k=1}^{K_l},\ \{\boldsymbol{h}_{t|l}^k\}_{k=1}^{K_l} = \text{SlotSSM}\Big(\{\boldsymbol{s}_{t|l}^k\}_{k=1}^{K_l},\ \{\boldsymbol{h}_{t-1|l}^k\}_{k=1}^{K_l}\Big) \tag{27}$$

$$\implies \quad \boldsymbol{y}_{t|l}^k, \boldsymbol{h}_{t|l}^k = \text{MambaBlock}\Big(\boldsymbol{s}_{t|l}^k, \boldsymbol{h}_{t-1|l}^k\Big)\,, \quad \forall k \in \{1, \ldots, K_l\} \tag{28}$$

Module	Hyperparameter	Dataset & Models	
		Blinking Color Balls (SlotSSMs)	MOVi-A (OC-SlotSSMs)
General	Batch Size	128	24
	Training Steps	300K	500K
	Sequence Length	{80, 160, 320, 640, 1024, 2048}	6
	Optimizer	AdamW	AdamW
	Weight Decay	0.1	0.1
	Learning Rate	8e-4	3e-4
Slot Encoder	Input Tokenizer	$\mathrm{MLP}(\mathrm{Patchify}(\boldsymbol{x}_{\text{input}}))$	$\mathrm{Flatten}(\mathrm{CNN}(\boldsymbol{x}_{\text{input}}))$
	Encoder Type	Self-Cross Attention	Inverted Attention
	Applied Layers	First Layer	All Layers
	Hidden Size	64	192
	Dropout	0	0
	Heads	4	4
SlotSSM	Hidden Size	64	192
	# Slots	6	11
	SSM Model	Mamba Block	Mamba Block
	State Size	16	16
	State Expand	1.25	1.25
Slot Mixer	Dropout	0	0
	Heads	4	4

Table 2: Hyperparameters of our model used in our experiments.

We include pseudo-code of the Mamba block implementation in Algorithm 1. For a more detailed description of the Mamba architecture and its underlying principles, we refer the readers to the original paper [24].

C.2 Baseline Models

We use the official implementation of RIM from GitHub [3], as well as the SAVi implementation from STEVE [4]. We describe the implementation of the proposed baselines SlotRNN and SlotTransformer in the following.

SlotRNN. SlotRNN adopts a similar design to SlotSSM, but replaces the SSMs with GRUs [5]. In this architecture, the slots are processed in parallel across different slots at each time step and sequentially across time steps. The implementation of each layer is summarized as follows.

$$\{\boldsymbol{s}_{t|l}^k\}_{k=1}^{K_l} = \mathrm{SlotEncoder}\Big(\{\boldsymbol{x}_{t|l-1}^k\}_{k=1}^{K_{l-1}}\Big)\ , \tag{29}$$

$$\boldsymbol{h}_{t|l}^k = \mathrm{GRU}\Big(\boldsymbol{s}_{t|l}^k, \boldsymbol{h}_{t-1|l}^k\Big), \quad \forall k \in \{1, \dots, K_l\}\ , \tag{30}$$

$$\{\boldsymbol{h}_{t|l}^k\}_{k=1}^{K_l} = \mathrm{SelfAttention}\Big(\{\boldsymbol{h}_{t|l}^k\}_{k=1}^{K_l}\Big)\ , \tag{31}$$

$$\{\boldsymbol{x}_{t|l}^k\}_{k=1}^{K_l} = \{\boldsymbol{h}_{t|l}^k\}_{k=1}^{K_l} \tag{32}$$

SlotTransformer. SlotTransformer uses the same SlotEncoder as SlotSSM to obtain slot representations. At each time step, the slots from the current step are concatenated with the slots from all previous time steps. This combined sequence is then processed using a Transformer with causal mask in time dimension which ensures that each slot can only obtain information from prior or current time steps. The implementation of each layer is summarized as follows:

[3] https://github.com/anirudh9119/RIMs

[4] https://github.com/singhgautam/steve

Table 3: The CNN encoder architecture used for object-centric learning.

Layer	Kernel Size	Stride	Padding	Channels	Activation
Conv	5×5	2	2	192	ReLU
Conv	5×5	1	2	192	ReLU
Conv	5×5	1	2	192	ReLU
Conv	5×5	1	2	192	None

Table 4: Spatial broadcast decoder architecture for image reconstruction in object-centric learning, it outputs RGB and alpha-mixing logits.

Layer	Kernel Size	Stride	Padding	Channels	Activation
Slot Normalization	-	-	-	-	-
Positional Embedding	-	-	-	-	-
ConvTranspose2d	5×5	2	2 (Output Padding: 1)	64	ReLU
ConvTranspose2d	5×5	2	2 (Output Padding: 1)	64	ReLU
ConvTranspose2d	5×5	2	2 (Output Padding: 1)	64	ReLU
ConvTranspose2d	5×5	2	2 (Output Padding: 1)	3 + 1	None

$$\{\boldsymbol{s}_{t|l}^k\}_{k=1}^{K_l} = \mathrm{SlotEncoder}\Big(\{\boldsymbol{x}_{t|l-1}^k\}_{k=1}^{K_{l-1}}\Big) \ , \tag{33}$$

$$\{\boldsymbol{x}_{<=t|l}^k\}_{k=1}^{K_l} = \mathrm{Transformer}\Big(\{\boldsymbol{s}_{t|l}^k\}_{k=1}^{K_l} \cup \{\boldsymbol{s}_{<t|l}^k\}_{k=1}^{K_l}\Big) \ , \tag{34}$$

C.3 Blinking Color Balls Experiemtns

We show the hyperparameters used in the experiments in Table 2.

Input Tokenizer. Each patch in the input sequence is treated as an image and further split into non-overlapping patches of size 4×4. Each patch is then augmented with spatial and temporal positional embeddings, followed by an MLP layer to compute the final tokens for the Slot Encoder.

Decoder. During image decoding, we use a self-cross attention layer with positional embeddings as input and slots as context. Given the positional embeddings $\mathcal{P}_t = \{\boldsymbol{p}_t^m\}_{m=1}^{HW}$ and slots from SlotSSM $\mathcal{S}_t = \{\boldsymbol{s}_t^k\}_{k=1}^{K}$, each layer of the transformer decoder can be described as follows:

$$\mathcal{P}_t = \mathrm{SelfAttn}(\mathcal{P}_t) \ , \tag{35}$$

$$\mathcal{P}_t = \mathrm{CrossAttn}\left(\mathtt{q} = \mathcal{P}_t,\ \mathtt{kv} = \mathcal{S}_t\right) \tag{36}$$

$$\mathcal{P}_t = \mathrm{MLP}(\mathcal{P}_t) \ . \tag{37}$$

We use a total of 3 layers, and the final pixel logits are computed using a linear head.

Training Objective. During training, we transform the image prediction problem into a pixel-wise classification task. Specifically, for a target image $\boldsymbol{x}_N \in \mathbb{R}^{H \times W \times 3}$, we compute a quantization by categorizing each pixel into one of 7 discrete color categories:

$$\boldsymbol{x}_N^Q(i,j) = Q(\boldsymbol{x}_N(i,j)) \quad \forall\, i \in \{1,2,\ldots,H\},\ j \in \{1,2,\ldots,W\} \tag{38}$$

where $Q : \mathbb{R}^3 \to \mathbb{C}$ is the quantization function that maps a 3-dimensional color vector to one of the 7 color categories in the set $\mathbb{C} = \{\boldsymbol{c}_1, \ldots, \boldsymbol{c}_7\}$. Each $\boldsymbol{c}_k \in \mathbb{R}^3$ represents a color vector corresponding to a discrete color category. This is a lossless quantization process since the raw images are generated with the same set of discrete colors. The final training objective is the cross-entropy loss between the model output $\hat{\boldsymbol{x}}_N$ and the target $\boldsymbol{x}_N^Q$:

$$\mathcal{L} = -\sum_{i=1}^{H}\sum_{j=1}^{W}\sum_{k=1}^{6} \boldsymbol{x}_N^{\mathrm{Q}}(i,j,k)\log(\hat{\boldsymbol{x}}_N(i,j,k)) \tag{39}$$

C.4 Unsupervised Object-Centric Learning Experiments

The hyperparameters used in the experiments are presented in Table 2. Table 4 details the structure of the spatial broadcast decoder described in Section 5.2.

To compute the input tokens, the input images are first processed by a CNN network to generate a 2D feature map. The architecture of the CNN network is described in Table 3. We use a downsampling factor of 2, resulting in an output 2D feature map of size 64×64 for an input image size of 128×128. The 2D feature map is then flattened into a sequence of length 4096 and provided to the inverted attention mechanism.

D Additional Results

D.1 Emerging Modularity in SlotSSMs

To gain further insights into the learned representations of the slot-based models, we investigate how the slots are utilized in the image generation process. This can be done by visualizing the attention mechanisms in the decoders.

Figure 12 presents the results of this analysis. For the transformer decoders used in the video prediction and blinking color balls tasks, we compute the argmax over the slots in the cross-attention map (Eq. 36), which represents the attention of the positional tokens over the slots employed to obtain information for reconstruction at each position. In the case of the spatial broadcast decoder, we take the argmax over the alpha-mixing logits $\boldsymbol{\alpha}_t$ (Eq. 17). The visualizations reveal that each slot tends to specialize in representing a specific object or a coherent part of the scene. This emerged object-centric representation allows the model to efficiently capture the dynamics and interactions of the objects, leading to improved performance in tasks such as video prediction and reasoning in the blinking color balls benchmark.

Interestingly, even though the slot encoder used in the video prediction and blinking color balls benchmarks does not explicitly enforce spatial disentanglement constraints like the inverted attention mechanism in OC-SlotSSMs does, the models still learn to represent the sequences in an object-centric manner. This emergent modularity suggests that the SlotSSM design can naturally encourages the model to discover and exploit the underlying structure of the data which is a crucial capability for modeling complex visual inputs such as real-world videos.

D.2 Real-World Videos Depth Estimation

In this additional experiment, we conduct a preliminary evaluation of SlotSSMs to assess its performance in real-world videos with increased visual complexity. Our aim is to observe how SlotSSMs utilize the modular representations to interpret and process real-world video data. Following previous works in object-centric learning [13], we evaluate this through a depth estimation task.

Datasets and Tasks. We select three datasets that represent distinct real-world application scenarios to observe the behavior of SlotSSMs across diverse contexts:

1. **TikTok Dancing Dataset** [33]: Commonly used for content creation tasks [31], this dataset comprises dynamic videos of individuals dancing with variety of movements and poses.
2. **UT Egocentric Video Dataset** [42]: Often utilized for egocentric action recognition [72], this dataset consists of first-person view videos that involve interactions with objects in the environment.
3. **Waymo Open Dataset** [59]: Primarily used for autonomous driving applications [43], this dataset includes videos captured from autonomous vehicles navigating traffic scenarios with diverse environmental conditions.

The primary task across these datasets is to estimate the depth of each pixel in the video frames. However, it is important to emphasize that our objective is not to develop depth estimation models that compete with existing specialized approaches. Instead, our main focus is to use this task, manageable with our lab resources, to showcase the emerging modularity in SlotSSMs for real-world video inputs.

Models. We compare OC-SlotSSM, which uses inverted attention in the Slot Encoder, against SAVi++ [13], an RNN-based object-centric learning method. Similar to the setting in Section 5, both models use a CNN encoder to extract input tokens, which are processed using inverted attention for OC-SlotSSM and slot attention for SAVi++ to produce slot representations. These slots are then used to reconstruct the image using a spatial broadcast decoder, with MSE loss as the training objective.

Results. The quantitative results in Table 5 show that OC-SlotSSM consistently outperforms the SAVi++ baseline across all datasets, demonstrating its superior video modeling capabilities. More importantly, as illustrated by the attention patterns in Figure 13, unsupervised scene decomposition emerges during training. This demonstrates that SlotSSM is able to utilize the modular representations to discover and exploit the latent structure of the input to complete the task, while the SAVi++ baseline does not demonstrate the same level of emergent modularity.

Table 5: Depth Estimation MSE (↓) on Real-World Datasets.

Model	UT Egocentric	Waymo	TikTok
SAVi++	0.589	0.804	1.412
OC-SlotSSM (Ours)	**0.464**	**0.653**	**1.180**

Algorithm 1 Mamba Block. The algorithm receives a T-length sequence of the same slot across time $\mathbf{s}_{1:T} \in \mathbb{R}^{T \times D}$. The algorithm outputs the updated slots $\mathbf{s}_{1:T}$. Note that the model imposes the diagonal structure on the $\mathbf{A}$ matrix.

Input: $\mathbf{s} \in \mathbb{R}^{T \times D}$
Block params: SSM linear $\mathtt{S}_B$, $\mathtt{S}_C$, $\mathtt{S}_\Delta$; Transition matrix $\mathbf{A} \in \mathbb{R}^{D \times N}$; LayerNorm `LN`; Linear $\mathtt{Linear}_1$, $\mathtt{Linear}_2$; 1D Conv `Conv1D`
  for $t = 1 \dots T$ **in parallel**
    $\mathbf{s}_t, \mathbf{res_t} = \mathtt{Linear}_1(\mathbf{s}_t)$
    $\mathbf{s}_t = \mathtt{SiLU}(\mathtt{Conv1D}(\mathbf{s}_t))$
  SSM block:
    $\mathbf{B} \in \mathbb{R}^{T \times N} \leftarrow \mathtt{S}_B(\mathbf{s})$
    $\mathbf{C} \in \mathbb{R}^{T \times N} \leftarrow \mathtt{S}_C(\mathbf{s})$
    $\Delta \in \mathbb{R}^{T \times D} \leftarrow \mathtt{SoftPlus}(\text{Parameter} + \mathtt{S}_\Delta(\mathbf{s}))$
    $\bar{\mathbf{A}}, \bar{\mathbf{B}} \in \mathbb{R}^{T \times D \times N} \leftarrow \text{discretize}(\Delta, \mathbf{A}, \mathbf{B})$
    $\mathbf{h}_0 = \mathbf{0}^{D \times N}$
    for $t = 1 \dots T$ **in parallel (scan)** # GPU hardware accelerated $\mathbf{y} \leftarrow \text{SSM}(\bar{A}, \bar{B}, C)(\mathbf{s})$.
      $\mathbf{h}_t = \bar{\mathbf{A}}_t \circ \mathbf{h}_{t-1} + \bar{\mathbf{B}}_t \mathbf{s}_t$ # Hadamard product for diagonal $\bar{\mathbf{A}}$.
      $\mathbf{y}_t = \mathbf{C}_t \mathbf{h}_t$
  for $t = 1 \dots T$ **in parallel**
    $\mathbf{y}_t = \mathbf{y}_t * \mathtt{SiLU}(\mathbf{res_t})$
    $\mathbf{y}_t = \mathtt{Linear}_2(\mathbf{y}_t)$
  return $\mathbf{y}$

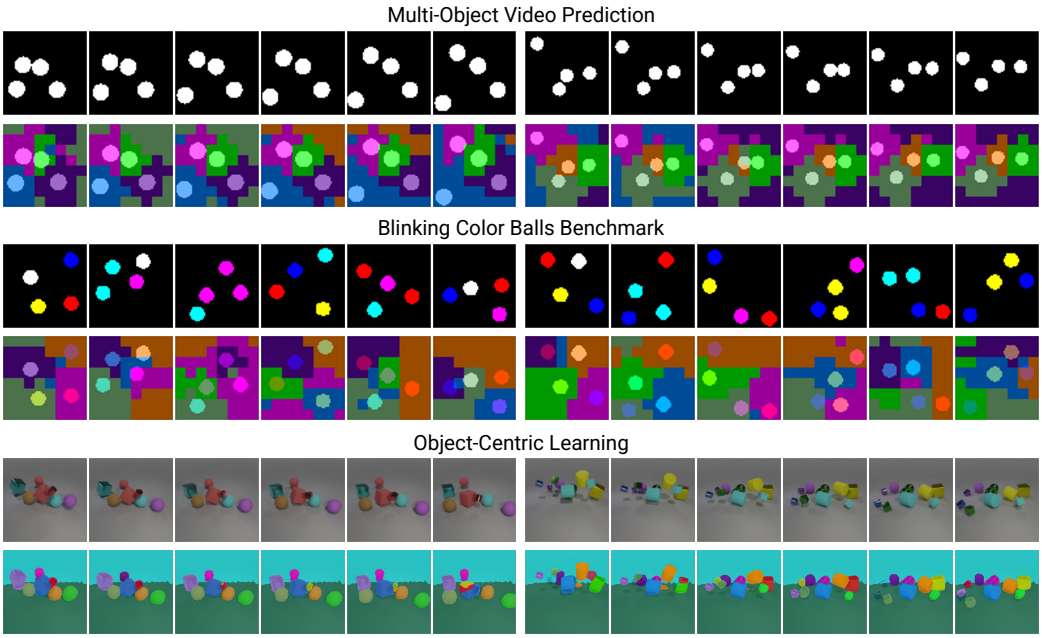


Figure 12: Emerging Modularity in SlotSSMs. Object-centric state representations naturally emerged to accommodate the underlying structure of the data.

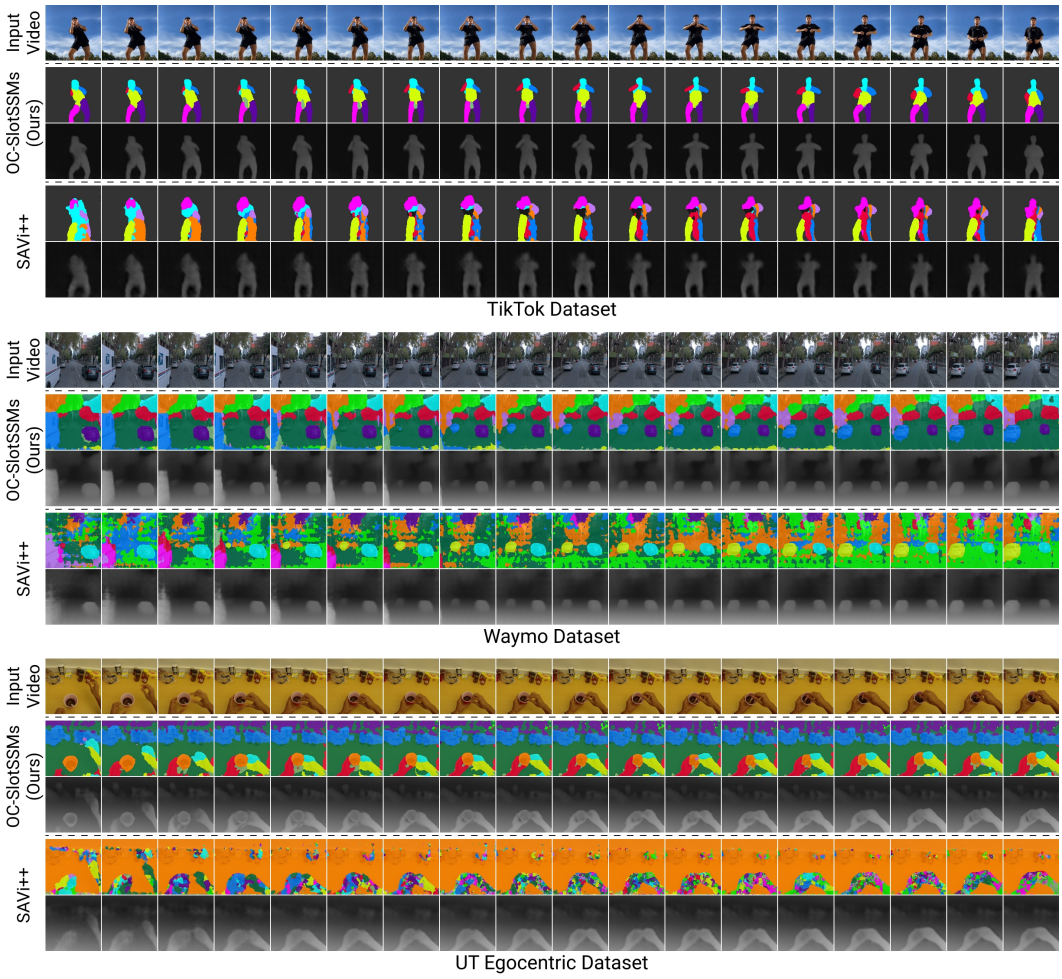


Figure 13: Emergent Scene Decomposition from Depth Estimation Tasks. Colors represent the ID of slots used for predicting each position. SlotSSM is capable of exploiting the inherent modular structure of real-world videos for efficient inference, without explicit segmentation supervision. For more examples please visit our project website.

NeurIPS Paper Checklist

The checklist is designed to encourage best practices for responsible machine learning research, addressing issues of reproducibility, transparency, research ethics, and societal impact. Do not remove the checklist: **The papers not including the checklist will be desk rejected.** The checklist should follow the references and follow the (optional) supplemental material. The checklist does NOT count towards the page limit.

Please read the checklist guidelines carefully for information on how to answer these questions. For each question in the checklist:

- You should answer [Yes] , [No] , or [NA] .
- [NA] means either that the question is Not Applicable for that particular paper or the relevant information is Not Available.
- Please provide a short (1–2 sentence) justification right after your answer (even for NA).

1. **Claims**

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

   Answer: [Yes]

   Justification: The abstract and introduction clearly state the main claims made in the paper, accurately reflecting its contributions and scope. The claims are supported by experimental results presented in the paper.

   Guidelines:

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

2. **Limitations**

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

   Answer: [Yes]

   Justification: Discussed in the section Conclusion and Limitations.

   Guidelines:

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

- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

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

   Answer: [NA]

   Justification: In this paper, we focus on a thorogh empirical study of the proposed architecture and do not include theoretical results.

   Guidelines:

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

4. **Experimental Result Reproducibility**

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

   Answer: [Yes]

   Justification: All the architectural details are explained in Section 4.

   Guidelines:

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

(a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
(b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
(c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

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

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

   Answer: [Yes]

   Justification: We will make our work open-source upon acceptance.

   Guidelines:

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

6. **Experimental Setting/Details**

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

   Answer: [Yes]

   Justification: Details are included in the Appendix and our open source code.

   Guidelines:

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

7. **Experiment Statistical Significance**

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

Answer: [No]

Justification: We did not include error bars in our reported results due to the prohibitively high computational cost of running multiple experimental trials. However, we have provided detailed experimental setups and parameters in the appendix. Additionally, we will open source our code to facilitate reproducibility and allow others to verify our findings.

Guidelines:

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

8. **Experiments Compute Resources**

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

   Answer: [Yes]

   Justification: Details are included in the Appendix and our open source code.

   Guidelines:

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

9. **Code Of Ethics**

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

   Answer: [Yes]

   Justification: We conducted in the paper conforms to the NeurIPS Code of Ethics in every respect.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.

- If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

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

    Answer: [Yes]

    Justification: We discussed this in the Impact Statement section.

    Guidelines:

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

11. **Safeguards**

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

    Answer: [Yes]

    Justification: Our work is not about image generators or pretrained language models and uses synthetic and non-risky datasets.

    Guidelines:

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

12. **Licenses for existing assets**

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

Answer: [Yes]

Justification: We used an existing dataset in our work and the original papers that produced the datasets. are cited in our paper.

Guidelines:

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

13. **New Assets**

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

Answer: [NA]

Justification: We do not release new assets.

Guidelines:

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

14. **Crowdsourcing and Research with Human Subjects**

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

Answer: [NA]

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

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

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

Answer: [NA]

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

Guidelines:

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