Compositional Visual Reasoning with SlotSSMs

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

Abstract

In many real-world sequence modeling problems, the underlying process is inherently modular and it is important to design machine learning architectures that can leverage this modular structure. In this paper, we introduce SlotSSMs, a novel framework for incorporating independent mechanisms into State Space Models (SSMs), such as Mamba, to preserve or encourage separation of information, thereby improving visual reasoning. We evaluate SlotSSMs on long-sequence reasoning, 3D visual reasoning, and real-world depth estimation tasks, demonstrating substantial performance improvements over existing sequence modeling methods. Our design efficiently exploits the modularity of inputs and scales effectively through the parallelizable architecture enabled by SSMs. We hope this approach will inspire future research on compositional reasoning architectures.

1 Introduction

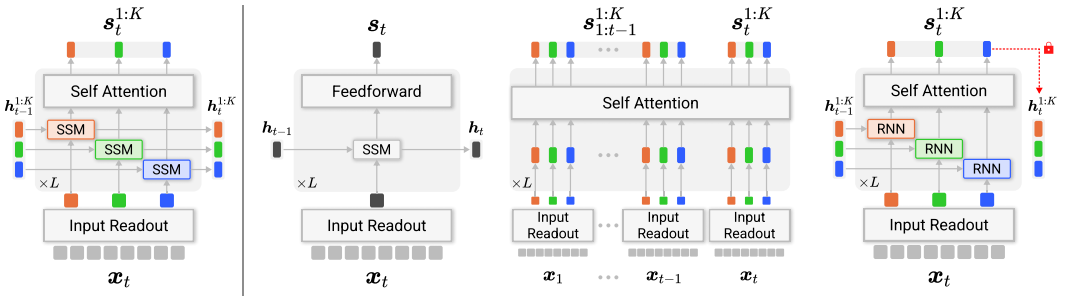
Recent progress in object-centric learning [23, 26, 18] has led to several methods for discovering modular object-centric structures and modeling their dynamics from videos with no or only weak supervision [20, 19, 5, 28]. Similar to RIMs [8], they build modularity into the RNN architecture to separately keep track of the dynamics of each object. However, RNNs are prone to vanishing gradients [25] 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. Unlike conventional SSMs that maintain a monolithic state vector, 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. 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.

In experiments, we evaluate SlotSSMs on visual reasoning and long-context reasoning tasks. By visualizing the decoder patterns, we reveal the emergent modularity in the model, demonstrating that SlotSSMs can successfully identify and leverage the modular structure of the input to complete the tasks.

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

NeurIPS 2024 Workshop on Compositional Learning: Perspectives, Methods, and Paths Forward.



(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.

2 Preliminary

A state space model (SSM) defines a sequence-to-sequence mapping between the input $\boldsymbol{e}_{1:T} \in \mathbb{R}^{T\times D}$ and the output $\boldsymbol{y}_{1:T} \in \mathbb{R}^{T\times D}$ by the following recurrence [13, 12, 29, 24]:

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

Here, T denotes the sequence length, $\boldsymbol{e}_t, \boldsymbol{y}_t \in \mathbb{R}^D$ are the input and output vectors at each time step t, and $\boldsymbol{h}_t \in \mathbb{R}^H$ is the hidden state that summarizes the history $\boldsymbol{e}_{\leq t}$. Recent works [9, 3] propose to employ learnable functions $\overline{\boldsymbol{A}}_t = \overline{\boldsymbol{A}}(\boldsymbol{e}_t)$, $\overline{\boldsymbol{B}}_t = \overline{\boldsymbol{B}}(\boldsymbol{e}_t)$, $\boldsymbol{C}_t = \boldsymbol{C}(\boldsymbol{e}_t)$. This brings the ability to selectively emphasize or ignore certain information based on the input.

3 Slot State Space Models (SlotSSMs)

SlotSSMs are fully parallelized sequential models consisting of three key components: the Slot Encoder, SlotSSM, and Slot Mixer, which we describe below.

3.1 SlotSSM

The core idea of SlotSSMs is to maintain separate *slot state* representations (or slots) and process them independently. This is achieved by making $\overline{\boldsymbol{A}}_t, \overline{\boldsymbol{B}}_t, \boldsymbol{C}_t$ block-diagonal, with each block conditioned on its corresponding slot:

$$\overline{\boldsymbol{A}}_t = \mathrm{diag}\left(\{\overline{\boldsymbol{A}}(\boldsymbol{s}_t^k)\}_{k=1}^K\right) , \quad \overline{\boldsymbol{B}}_t = \mathrm{diag}\left(\{\overline{\boldsymbol{B}}(\boldsymbol{s}_t^k)\}_{k=1}^K\right) , \quad \boldsymbol{C}_t = \mathrm{diag}\left(\{\boldsymbol{C}(\boldsymbol{s}_t^k)\}_{k=1}^K\right) . \tag{2}$$

Next, we complement SlotSSM with a slot encoder that extracts slot representations from unstructured inputs (Section 3.2), and a slot mixer that introduces sparse interactions across slots (Section 3.3).

3.2 Slot Encoder

We represent the unstructured input $\boldsymbol{x}_t$ at each time step as a sequence of M tokens, $\boldsymbol{x}_t = (\boldsymbol{x}_t^1, \ldots, \boldsymbol{x}_t^M)$. For example, image inputs may be represented as CNN feature maps or embeddings of non-overlapping image patches, as in ViT [4]. Slot representations $\{\boldsymbol{s}_t^k\}_{k=1}^K$ are extracted via cross-attention using a Transformer [33]:

$$\{\boldsymbol{s}_t^k\}_{k=1}^K \leftarrow \mathrm{Transformer}\big(\mathtt{q} = \{\boldsymbol{s}_t^k\}_{k=1}^K,\ \mathtt{kv} = \{\boldsymbol{x}_t^m\}_{m=1}^M\big) \ . \tag{3}$$

The learned queries, $\{\boldsymbol{s}_t^k\}_{k=1}^K$, allow the Transformer to facilitate the emergence of modularity by capturing information across different input regions. Alternatively, the Transformer can be replaced with inverted attention [32, 34], which is cross attention with Softmax over queries instead of keys. This design softly assigning each input token to a slot, thereby promoting modularity, resulting in our varient Object-Centric SlotSSMs (OC-SlotSSMs).

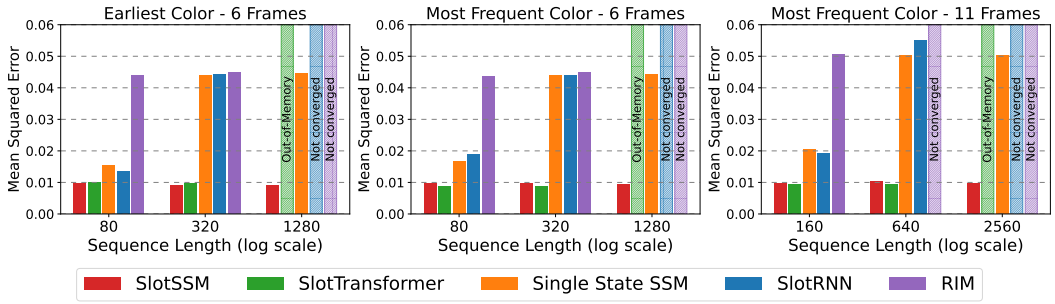


Figure 2: Long-Context Reasoning in the Blinking Balls Benchmark. SlotSSM maintains consistent performance across sequence lengths from 80 to 2560, while baseline models show degraded performance or fail to complete training due to high memory and computational demands.

3.3 Slot Mixer

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 which consists of self-attention [33] and MLP layers. Note that the output of Slot Mixer carries information from the entire history of each slot, it provides the opportunity to refine the slot representations based on temporal dynamics.

4 Related Work

State Space Models (SSMs). SSMs, popularized by S4 [12], have gained significant attention in language and sequence modeling. S4 leverages HiPPO theory [10] to parameterize state transition matrices, though this approach is mathematically complex. Recent works have simplified SSMs with diagonal transition matrices [14, 11, 29] and RNN-based formulations without relying on ODE discretization [15, 24, 3]. Mamba [9] introduced a more flexible approach by conditioning SSM parameters on inputs, improving adaptability in sequence modeling. **Object-Centric Learning.** Object-centric learning aims to discover modular representations [1, 6, 22, 17, 19, 8], such as objects and their interactions, from images and videos with minimal supervision. Slot Attention [23] has become a dominant model in this area, utilizing GRUs [2] and competitive attention mechanisms to iteratively refine slot representations [26, 28, 20, 5, 35, 27, 18, 36]. However, RNN-based methods face limitations such as vanishing gradients and the inability to parallelize training. In contrast, our SlotSSMs, built on SSMs, enable parallel training and exhibit strong long-term memory capabilities.

5 Experiments

5.1 Long-Context Video Reasoning

We evaluate the model's ability to handle long-context reasoning using the Blinking Color Balls Benchmark, a novel task designed to test sequence modeling over extended time frames.

Blinking Color Balls Benchmark. This benchmark consists of video sequences $\mathbf{x}_{1:T}$ with bouncing balls. The sequences are divided into context frames $\mathbf{x}_{1:T-1}$ and a target frame $\mathbf{x}_T$. Initially, all balls are white. At each timestep in $\mathbf{x}_{1:T-1}$, one ball is randomly assigned a non-white color. The task is to predict the color of each ball in the target frame based on two variants: (1) *Earliest Color*: each ball retains the earliest non-white color assigned to it; and (2) *Most Frequent Color*: each ball is colored according to the most frequent non-white color it received across the context frames. Balls that remain white in all context frames stay white in the target frame.

To convert the task into a long-range reasoning problem, we patchify each context image into non-overlapping patches and provide the flattened sequence of patches as input. We use $T = 6$ for the Earliest Color variant and $T \in \{6, 11\}$ for the Most Frequent Color variant, with patch sizes of $P \in \{4, 8, 16\}$, resulting in input sequence lengths $L \in \{80, 160, 320, 640, 1280, 2560\}$. The task for the models is to generate the target image based on this long sequential input.

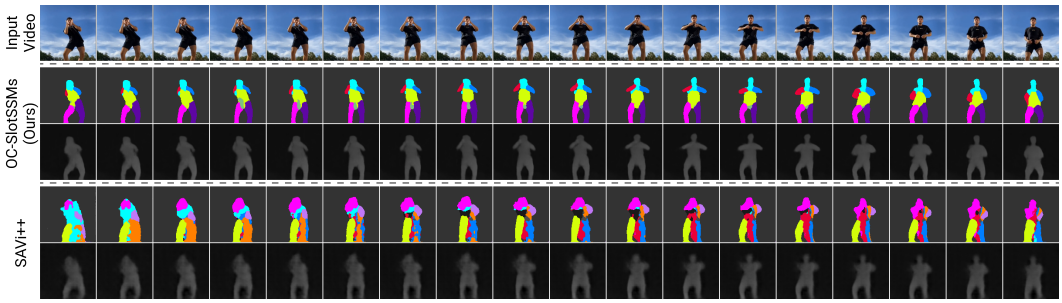


Figure 3: Emergent Scene Decomposition from Depth Estimation Tasks. We use the TikTok dataset as an example. Colors represent the ID of slots used for predicting each position. SlotSSM is capable of exploiting the inherent modular structure of real-world videos for efficient inference, without explicit segmentation supervision.

Models. We compare SlotSSM to several baselines: (1) Single State SSM, which shares the same architecture as SlotSSM but uses a single, monolithic state; (2) SlotTransformer, a Transformer model with multiple slots at each timestep; (3) RIM [8], and (4) SlotRNN, a variant of RIM with shared RNN weights across slots and dense state updates. All models share the same Transformer-based image decoder for a fair comparison.

Results. The results, shown in Figure 2, highlight SlotSSM's clear advantages over the baselines across all sequence lengths. While Single State SSM and SlotRNN perform well on shorter sequences, their performance deteriorates significantly for sequences longer than 320 frames. Surprisingly, RIM fails at all sequence lengths, likely due to optimization challenges arising from separate weights for each slot. SlotRNN partially addresses this issue by sharing weights across slots while preserving modularity. SlotTransformer performs well up to 640 frames, benefiting from direct access to all historical inputs. However, SlotSSM excels in long-range reasoning, particularly on sequences of 1280 and 2560 frames, where other models either fail to train or face severe memory constraints.

5.2 3D Visual Reasoning

Finally, we explore the application of SlotSSM and OC-SlotSSM to 3D visual reasoning tasks using the CATER benchmark [7].

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

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**

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.

5.3 Real-World Depth Estimation

In this experiment, we aim to observe how SlotSSMs utilize the modular representations to interpret and process real-world video data. Following previous works in object-centric learning [5], 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) the UT Egocentric video dataset [21], (2) the Waymo autonomous driving video dataset [30], and (3) the TikTok dancing dataset [16].

The primary task is to estimate the depth of each pixel in the video frames. However, it is important to emphasize that 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++ [5], an RNN-based object-centric learning method. Both models use a CNN encoder to extract input tokens, which are processed using their respective attention mechanisms (inverted attention for OC-SlotSSM and slot attention for SAVi++) to produce slot representations. These slots are then used to reconstruct the image and generate object segmentation masks using a spatial broadcast decoder, with reconstruction as the training objective.

Results. The quantitative results in Table 2 show that OC-SlotSSM consistently outperforms the SAVi++ baseline across all datasets, demonstrating its superior video modeling capabilities. Moreover, the attention patterns in Figure 3 reveal that unsupervised scene decomposition emerges during training without any segmentation supervision, highlighting OC-SlotSSM's ability to leverage the inherent modular structure of real-world videos for downstream tasks.

Table 2: Depth Estimation MSE (↓) on Different Datasets.

Model	UT Egocentric	Waymo	TikTok
SAVi++	0.5885	0.804	1.412
OC-SlotSSM (Ours)	**0.4640**	**0.653**	**1.180**

6 Conclusion

We introduced SlotSSMs, a novel approach to incorporating modular structures and inductive biases into State Space Models for improved sequence modeling. By maintaining independent slot vectors and performing state transitions independently with sparse interactions via self-attention, SlotSSMs effectively capture the modularity inherent in real-world processes. Our experiments demonstrate significant performance improvements over existing sequence modeling methods.

Acknowledgements

This work is supported by Brain Pool Plus Program (No. 2021H1D3A2A03103645) and Young Researcher Program (No. 2022R1C1C1009443) through the National Research Foundation of Korea (NRF) funded by the Ministry of Science and ICT. This research was supported by GRDC (Global Research Development Center) Cooperative Hub Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science and ICT (MSIT) (RS-2024-00436165).

References

[1] 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.

[2] 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.

[3] 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.

[4] 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.

[5] 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.

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

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

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

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

[10] 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.

[11] 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.

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

[13] 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.

[14] 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.

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

[16] 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.

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

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

[19] 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.

[20] 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.

[21] 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.

[22] 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.

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

[24] 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.

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

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

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

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

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

[30] 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.

[31] 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.

[32] 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.

[33] 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.

[34] 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.

[35] 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.

[36] 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.

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, although the proposed architecture is not only applicable to video modeling but also to other modalities like text, we have not explored this aspect in this study. 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. Lastly, future studies should investigate the effect of increased visual complexity in videos. To this end, in Appendix **??**, we present a preliminary study applying SlotSSMs to natural videos, demonstrating how modularity emerges from SlotSSMs' independent mechanisms in real-world scenes. 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.

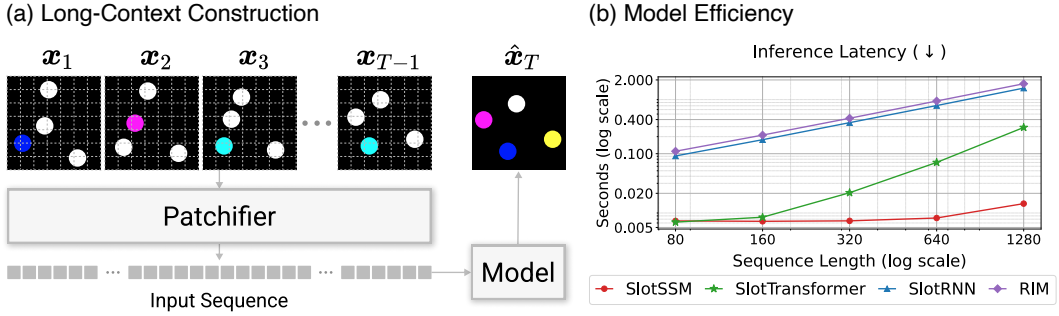


Figure 4: Long-Context Construction and Model Efficiency in the Blinking Color Balls Benchmark. *Left*: We construct long-sequence inputs by patchifying the context images. *Right*: Comparison of model inference latency with batch size 6. SlotSSM demonstrates computational efficiency for long-sequence processing tasks.

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 [31], 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.

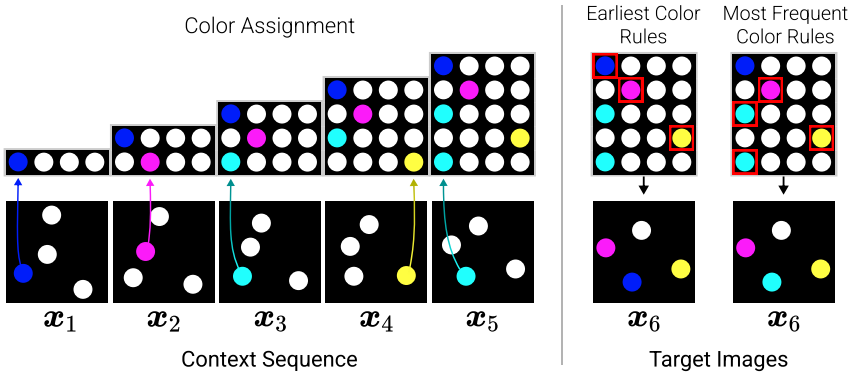


Figure 5: Blinking Color Balls Benchmark Overview. *Left*: Context frames with independent random ball picking and color assignments for each frame. Top figures indicate the sequential color assignment. *Right*: Target image for the Earliest Color and Most Frequent Color variants. Top figures indicate the color assignment rules.

B.2 Dataset Design

We provide an illustrative example of the dataset design in Figure 5. 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 5(top). Note that the object picking and coloring are performed independently for each timestep, thus one ball could 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 5, 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 6.

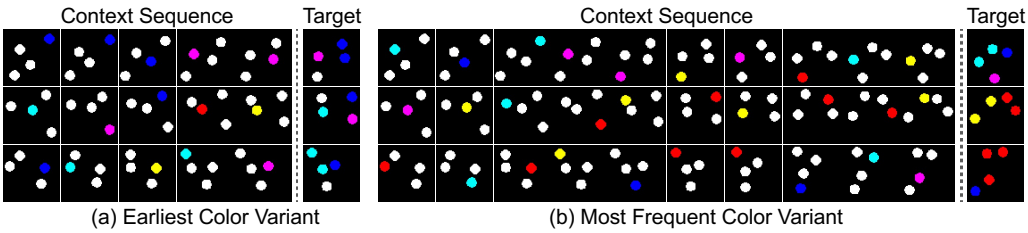


Figure 6: Blinking Color Balls Samples.

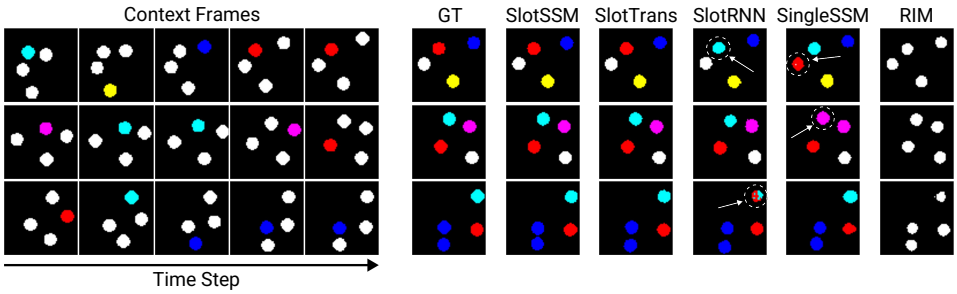


Figure 7: Blinking Color Balls Qualitative Comparison. Results shown for the Most Frequent Color variant with a sequence length of 80 frames.

Finally, as illustrated in Figure 4(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 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 7 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 8. 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:

$$\mathcal{C}_t = \mathrm{SelfAttn}(\mathcal{C}_t)\ , \tag{4}$$

$$\mathcal{C}_t = \mathrm{CrossAttn}\left(\mathtt{q} = \mathcal{C}_t,\ \mathtt{kv} = \mathcal{X}_t\right)\ , \tag{5}$$

$$\mathcal{C}_t = \mathrm{MLP}(\mathcal{C}_t)\ . \tag{6}$$

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:

$$Q = W_Q(\mathcal{C}_t), \quad K = W_K(\mathcal{X}_t), \quad V = W_V(\mathcal{X}_t)\,, \tag{7}$$

$$\mathcal{C}_t^{\text{out}} = \texttt{softmax}\left(\frac{QK^T}{\sqrt{D}}, \quad \texttt{axis='keys'}\right) V\,. \tag{8}$$

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:

$$Q = W_Q(\mathcal{C}_t), \quad K = W_K(\mathcal{X}_t), \quad V = W_V(\mathcal{X}_t)\,, \tag{9}$$

$$A = \texttt{softmax}\left(\frac{QK^T}{\sqrt{D}}, \quad \texttt{axis='queries'}\right)\,, \tag{10}$$

$$A_{i,j} = \frac{A_{i,j}}{\sum_{j=1}^{N_K} A_{i,j}}, \tag{11}$$

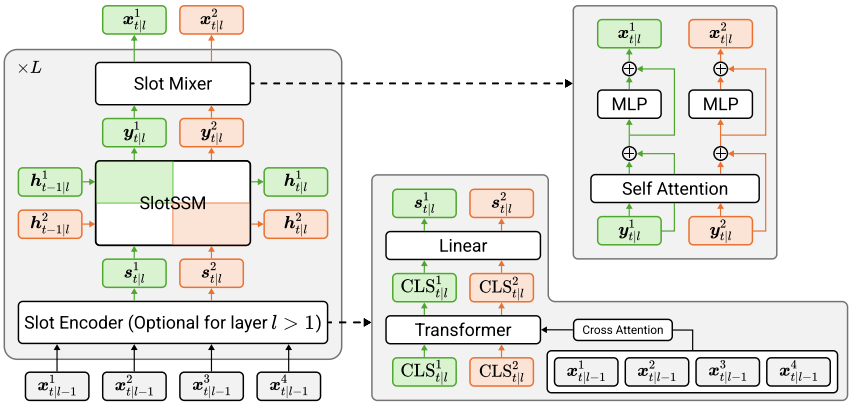
$$\mathcal{C}_t^{\text{out}} = AV\,. \tag{12}$$

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.

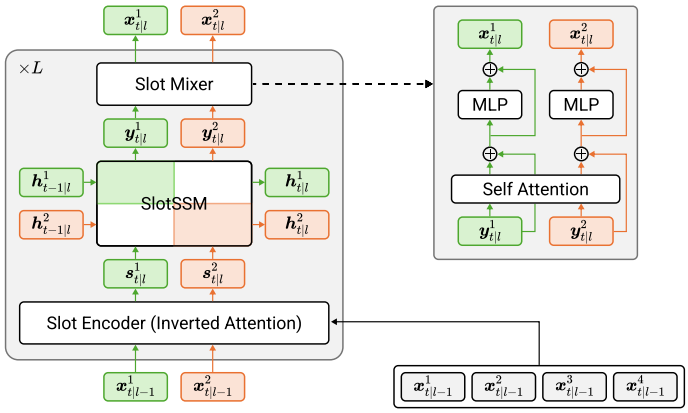
		Dataset & Models	
Module	Hyperparameter	Blinking Color Balls (SlotSSMs)	MOVi-A (OC-SlotSSMs)
General	Batch Size	128	24
	Training Steps	300K	500K
	Sequence Length	{80, 160, 320, 640, 1024, 2048}	6
	Optimizer	AdamW	AdamW
	Weight Decay	0.1	0.1
	Learning Rate	8e-4	3e-4
Slot Encoder	Input Tokenizer	$\mathrm{MLP}(\mathrm{Patchify}(\boldsymbol{x}_{\text{input}}))$	$\mathrm{Flatten}(\mathrm{CNN}(\boldsymbol{x}_{\text{input}}))$
	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 3: Hyperparameters of our model used in our experiments.

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 [9]. The block-diagonal transition of slots is implemented as parallel runs of SSM blocks that share the same model weights.



(a) SlotSSMs



(b) OC-SlotSSMs

Figure 8: SlotSSMs vs OC-SlotSSMs.

$$\{\boldsymbol{y}^k_{t|l}\}_{k=1}^{K_l},\ \{\boldsymbol{h}^k_{t|l}\}_{k=1}^{K_l} = \text{SlotSSM}\Big(\{\boldsymbol{s}^k_{t|l}\}_{k=1}^{K_l},\ \{\boldsymbol{h}^k_{t-1|l}\}_{k=1}^{K_l}\Big) \tag{13}$$

$$\implies \quad \boldsymbol{y}^k_{t|l}, \boldsymbol{h}^k_{t|l} = \text{MambaBlock}\Big(\boldsymbol{s}^k_{t|l}, \boldsymbol{h}^k_{t-1|l}\Big), \quad \forall k \in \{1, \dots, K_l\} \tag{14}$$

We include pseudo-code of the Mamba block implementation in Algorithm **??**. For a more detailed description of the Mamba architecture and its underlying principles, we refer the readers to the original paper [9].

C.2 Baseline Models

We use the official implementation of RIM from GitHub [2], as well as the SAVi implementation from STEVE [3]. 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 [2]. 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.

[2] https://github.com/anirudh9119/RIMs
[3] https://github.com/singhgautam/steve

$$\{\boldsymbol{s}_{t|l}^k\}_{k=1}^{K_l} = \text{SlotEncoder}\Big(\{\boldsymbol{x}_{t|l-1}^k\}_{k=1}^{K_{l-1}}\Big) \ , \tag{15}$$

$$\boldsymbol{h}_{t|l}^k = \text{GRU}\Big(\boldsymbol{s}_{t|l}^k, \boldsymbol{h}_{t-1|l}^k\Big), \quad \forall k \in \{1, \dots, K_l\} \ , \tag{16}$$

$$\{\boldsymbol{h}_{t|l}^k\}_{k=1}^{K_l} = \text{SelfAttention}\Big(\{\boldsymbol{h}_{t|l}^k\}_{k=1}^{K_l}\Big) \ , \tag{17}$$

$$\{\boldsymbol{x}_{t|l}^k\}_{k=1}^{K_l} = \{\boldsymbol{h}_{t|l}^k\}_{k=1}^{K_l} \tag{18}$$

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:

$$\{\boldsymbol{s}_{t|l}^k\}_{k=1}^{K_l} = \text{SlotEncoder}\Big(\{\boldsymbol{x}_{t|l-1}^k\}_{k=1}^{K_{l-1}}\Big) \ , \tag{19}$$

$$\{\boldsymbol{x}_{<=t|l}^k\}_{k=1}^{K_l} = \text{Transformer}\Big(\{\boldsymbol{s}_{t|l}^k\}_{k=1}^{K_l} \cup \{\boldsymbol{s}_{<t|l}^k\}_{k=1}^{K_l}\Big) \ , \tag{20}$$

C.3 Blinking Color Balls Experiemtns

We show the hyperparameters used in the experiments in Table 3.

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:

$$\mathcal{P}_t = \text{SelfAttn}(\mathcal{P}_t) \ , \tag{21}$$

$$\mathcal{P}_t = \text{CrossAttn}\left(\texttt{q} = \mathcal{P}_t, \ \texttt{kv} = \mathcal{S}_t\right) \tag{22}$$

$$\mathcal{P}_t = \text{MLP}(\mathcal{P}_t) \ . \tag{23}$$

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:

$$\boldsymbol{x}_N^Q(i,j) = Q(\boldsymbol{x}_N(i,j)) \quad \forall \ i \in \{1, 2, \dots, H\}, \ j \in \{1, 2, \dots, W\} \tag{24}$$

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 $\mathbb{C} = \{\boldsymbol{c}_1, \dots, \boldsymbol{c}_7\}$. 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 $\boldsymbol{x}_N^Q$:

$$\mathcal{L} = -\sum_{i=1}^{H}\sum_{j=1}^{W}\sum_{k=1}^{6} \boldsymbol{x}_N^Q(i,j,k) \log(\hat{\boldsymbol{x}}_N(i,j,k)) \tag{25}$$