Unsupervised Multi-Object Discovery and Tracking using Memory-Augmented Slot Attention

by

Ahmed Imam Shah

A Dissertation Submitted to the
Graduate School of Sciences and Engineering
in Partial Fulfillment of the Requirements for
the Degree of

Master of Science

in

Computer Science and Engineering

September 11, 2023

Unsupervised Multi-Object Discovery and Tracking using Memory-Augmented Slot Attention

Koç University

Graduate School of Sciences and Engineering

This is to certify that I have examined this copy of a master's thesis by

Ahmed Imam Shah

and have found that it is complete and satisfactory in all respects,
and that any and all revisions required by the final
examining committee have been made.

Committee Members:

_____

Prof. Yücel Yemez (Advisor)

_____

Prof. Engin Erzin

_____

Assist. Prof. Furkan Kıraç

_____

Assoc. Prof. Aykut Erdem (Co-Advisor)

Date: _____

Dedicated to my parents, whose unconditional support made this possible.

ABSTRACT

Unsupervised Multi-Object Discovery and Tracking using Memory-Augmented Slot Attention
Ahmed Imam Shah
Master of Science in Computer Science and Engineering
September 11, 2023

Learning object-centric representations from static images is a promising research direction in the field of deep learning. However, adapting this approach to videos poses certain challenges due to the necessity of capturing the temporal dynamics of video content. Recent works have made significant progress in object discovery within synthetic video datasets. Nevertheless, these works do not fully exploit the motion of objects in videos and temporal cues. In this thesis, we aim to enhance the performance of object-centric representation learning on video frames by using the temporal information more carefully. To achieve this goal, we propose a new unsupervised learning method that utilizes a memory-augmented slot attention model for multi-object discovery and tracking. The key component of our approach is the integration of memory slots, which store information from past video frames, alongside object slots into the learning architecture. Object slots simultaneously attend to both memory slots for information from past frames and the current image input. Training memory slots requires longer video sequences. However, the most suitable learning structures for this, recurrent neural networks (RNNs), are not very effective in learning long-term temporal data due to the issues of exploding and/or vanishing gradients. To train our memory-augmented model more effectively on long video sequences, we employ truncated back-propagation through time. Experiments conducted on synthetic yet realistic video datasets have yielded promising results, indicating that memory slots significantly improve multi-object tracking and object segmentation performance. Our fully unsupervised learning method contributes to the problem of object-centric representation learning in videos and opens up new possibilities in this field.

ÖZETÇE

Bellek Destekli Slot Dikkat Modeliyle Gözetimsiz Çoklu Nesne Keşfi ve Takibi

Ahmed Imam Shah

Bilgisayar Bilimi ve Mühendisliği, Yüksek Lisans

11 Eylül 2023

Durağan görüntülerden nesne-merkezli temsil öğrenme, derin öğrenme alanında umut vadeden bir yaklaşımdır. Ancak bu yaklaşımı hareketli görüntülere uyarlamak, video içeriğinin zamansal dinamiklerini yakalama gerekliliği nedeniyle bazı zorluklar içermektedir. Yakın zamanlı çalışmalar, sentetik video veri kümelerinde nesne keşfi konusunda bazı önemli ilerlemeler kaydetmiştir. Bununla birlikte, bu çalışmalarda videolardaki nesnelerin hareketleri ve zamanla ilgili ipuçları tam olarak dikkate alınmazlar. Bu tez çalışmasında, bu bilgilerin daha dikkatli kullanımıyla video görüntüleri üzerindeki nesne-merkezli temsil öğrenme başarımının arttırılması hedeflenmektedir. Bu amaca yönelik olarak, çoklu nesne keşfi ve takibi için bellek destekli slot dikkat modelini kullanan yeni bir gözetimsiz öğrenme yöntemi öneriyoruz. Yaklaşımımızın anahtarı, nesne slotlarına ek olarak öğrenme mimarisine entegre ettiğimiz ve video çerçevelerinden bilgi depolayan bellek slotlarıdır. Nesne slotları, geçmiş çerçevelerden bilgi almak için eş zamanlı olarak hem bellek slotlarına hem de o andaki görüntü girdisine dikkat ederler. Bellek slotlarının uzun video dizileri üzerinde eğitilmeleri gerekir. Ne var ki, bunun için en uygun öğrenme yapıları olan yinelemeli sinir ağları (RNN), patlayan ve/veya kaybolan gradyan problemine bağlı olarak, uzun süreli zamansal verileri öğrenmede çok etkili olamazlar. Bellek destekli modelimizi uzun video dizileri üzerinde RNN yapılarını kullanarak daha etkili bir şekilde eğitebilmek için, zamanda kısaltılmış geri yayılım tekniğini kullanıyoruz. Sentetik ancak gerçekçi görüntüler içeren video veri kümeleri üzerinde gerçekleştirdiğimiz deneyler umut verici sonuçlar üretmiştir; bu sonuçlar bellek slotlarının çoklu nesne takibi ve nesne bölütleme başarımını önemli ölçüde artırdığını göstermektedir. Tamamen gözetimsiz olarak çalışan öğrenme yöntemimiz, videolar üzerinde nesne-merkezli temsil öğrenme problemine katkı sağlamakta ve bu alanda yeni olanakların yolunu açmaktadır.

ACKNOWLEDGMENTS

I would like to express my heartfelt gratitude to my thesis advisors, Prof. Yücel Yemez and Assoc. Prof. Aykut Erdem, for their invaluable guidance, support, and expertise throughout my research journey. Their insights and encouragement have been instrumental in shaping this work.

I am deeply thankful to KUIS AI Center for awarding me the AI Fellowship, which provided the necessary funding for my research and connected me with a community of fellow researchers. I am also grateful to Koç University for providing a supportive academic environment that nurtured my growth as a researcher.

I would like to extend my appreciation to my peers, Can Küçüksözen and Abdul Basit Anees, for their insights and assistance during the research process. Their contributions have been truly invaluable.

To my family and friends, I am forever grateful for your motivation and belief in my abilities. Your positivity has made this journey even more meaningful.

A special note of thanks goes to my wife for her boundless patience, unwavering support, and understanding during the ups and downs of this journey. Your encouragement and belief in me have been my driving force.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABBREVIATIONS

ARI	Adjusted Rand Index
CE	Cross-Entropy
CNN	Convolutional Neural Network
dVAE	Discrete Variational Autoencoder
FG-ARI	Foreground - Adjusted Rand Index
GRU	Gated Recurrent Unit
LN	Layer Normalization
ML	Mostly Lost
MLP	Multi-Layer Perceptron
MT	Mostly Tracked
MOT	Multi-Object Tracking
MOTA	Multi-Object Tracking Accuracy
MSE	Mean Squared Error
RNN	Recurrent Neural Network
TBPTT	Truncated Backpropagation Through Time

Chapter 1

INTRODUCTION

Humans comprehend the world in terms of separate objects [Kahneman et al., 1992; Spelke and Kinzler, 2007]. We effortlessly distinguish different objects and the background, successfully identify stationary and moving objects, and even predict the motion of the moving objects. Object-centric representation learning draws inspiration from human perception and focuses on unraveling the modular and causal arrangement within visual inputs. This approach is frequently facilitated through specialized neural network architectures that excel in unsupervised object-centric representation learning. These models are capable of significantly enhancing the efficiency of sampling, robustness, adaptability to novel tasks, and interpretability of machine learning algorithms [Greff et al., 2020].

The pursuit of object-centric representation learning from still images has emerged as a promising direction, leading to advancements in tasks such as object discovery and segmentation. Locatello et al. [2020] proposed object-centric learning through the so-called slot attention models, providing insights into the fundamentals of iterative attention through slot attentions and decomposing a scene into slots. Singh et al. [2022a] explored the innovative application of generative models for image composition and changed the mixture-based decoder to a transformer-based decoder.

Nevertheless, extending this technique to videos is challenging, as it entails capturing the temporal dimension inherent in the video content. Recent works have shown significant progress in object discovery using synthetic video datasets. The work of Kipf et al. [2021] uses the slot attention model of Locatello et al. [2020] and applies it recurrently to the videos. Furthermore, Singh et al. [2022b] have presented a simple yet effective method for unsupervised object-centric learning from intricate

and lifelike videos. However, these works do not fully utilize the objects' motion and temporal cues in the videos, which can be used to improve the performance further.

This thesis addresses this shortcoming by introducing an innovative strategy that employs slot attention models in the video domain. The key to our approach is introducing memory slots, which store information from past frames. They are responsible for remembering and retaining the objects' information. These models utilize memory slots to propagate information across the frames, facilitating a more comprehensive understanding of object interactions over time. This also helps in consistently tracking the objects over the video sequence. We measure the performance of our model in terms of object segmentation using Adjusted Rand Index Rand [1971]. Moreover, we extend our experimentation to test how our model works for multi-object tracking tasks. We track objects over the sequence of frames and evaluate their performance based on the Multi-Object Tracking (MOT) metrics, given in [Bernardin and Stiefelhagen, 2008].

In the following sections of this thesis, we will delve into the theoretical foundations of our proposed methodology, explain its technical details, and analyze experimental evaluations. Through a comprehensive review of our approach, we aspire to highlight the effectiveness of our Memory Augmented Slot Attention for Video model in facilitating unsupervised object-centric representation learning in the dynamic context of video data. By bridging the gap between object-centric representation and video analysis, this research contributes to the advancement of computer vision.

Chapter 2

RELATED WORK AND BACKGROUND

2.1 Unsupervised Object-Centric Representation Learning in Images

Object-centric representation learning aims to discover the underlying structures and objects in the visual data and represent them in a set of feature vectors called slots. In recent years, unsupervised or self-supervised object-centric representation learning methods for still images have received significant interest from researchers [Greff et al., 2016; Eslami et al., 2016; Greff et al., 2017; Burgess et al., 2019; Greff et al., 2019; Engelcke et al., 2019; Crawford and Pineau, 2019; Löwe et al., 2020; Lin et al., 2020b; Anciukevicius et al., 2020; Locatello et al., 2020; Zoran et al., 2021; Caron et al., 2021; Chen et al., 2021; Wang et al., 2022]. These models are generally trained by using reconstruction loss. Usually, in these works, a mixture-based decoder is used to render each slot's contents independently and reconstruct the image. Some researchers have pursued another direction of minimizing the mutual information between the predicted object segmentations [Yang et al., 2020; Savarese et al., 2021]. Lately, energy-based models are used for object-centric representation learning [Du et al., 2021; Yu et al., 2021]. Most recently, Löwe et al. used complex-valued neural networks for scene decomposition. Some works have shown the combination of a slot-attention encoder with a transformer-based decoder which can also be used for object-centric learning [Singh et al., 2022a; Lamb et al., 2021]. By using transformer-based decoders, realistic-looking images could also be decomposed into slots.

To better express our contributions and provide a comprehensive understanding of our work, we will thoroughly review the fundamentals of two image-based models: Locatello et al. [2020] and Singh et al. [2022a].

2.1.1 Slot Attention Autoencoder

In 2020, Locatello et al. presented a seminal paper titled "Object-centric Learning with Slot Attention." This model was designed for synthetic still images and was the first time the notion of slot attention was introduced. In the first step of the algorithm, a CNN-based encoder extracts the features from the image $\mathbf{x}$. These features are normalized using the Layer Normalization [Ba et al., 2016]. A position embedding is added to the features, and these features, $\mathbf{h}$, are then passed as input to the slot attention module. The slots $\mathcal{S}$ are randomly initialized from the Gaussian distribution with learnable mean and variance:

$$\mathcal{S} \sim \mathcal{N}(\mu, \sigma) \tag{2.1}$$

where, $\mathcal{S} = [\mathcal{S}_1, \mathcal{S}_2, \ldots, \mathcal{S}_L]$ and L is the total number of slots. The slot attention module uses dot-product attention [Luong et al., 2015] with attention coefficients $\mathrm{A}_{i,j}$ that are normalized over queries, in this case, slots:

$$\mathrm{A}_{i,j} = \frac{e^{B_{i,j}}}{\sum_l e^{B_{i,l}}} \tag{2.2}$$

where,

$$\mathbf{B} = \frac{1}{\sqrt{D}} k(\mathbf{h}) \cdot q(\mathcal{S})^T \in \mathbb{R}^{N \times L} \tag{2.3}$$

Here, k is the linear projection applied to $\mathbf{h}$, which is then used as the key, and q is the linear projection applied to slots $\mathcal{S}$ which becomes the query in the attention computation. N is the dimension of the input features $\mathbf{h}$, and D is the slot size. The softmax function in Equation 2.2 is used over the slots axis to compute the attention maps $\mathbf{A}_j$, $j = 1 \ldots L$. Moreover, $\frac{1}{\sqrt{D}}$ is used as the temperature constant for the Softmax operation.

In order to aggregate the input values to their respective slots, another linear projection v is applied on $\mathbf{h}$ and multiplied by the transpose of $\mathbf{W}$:

$$\mathbf{U} = \mathbf{W}^T \cdot v(\mathbf{h}) \in \mathbb{R}^{K \times D} \tag{2.4}$$

where $\mathbf{W}$ is the weight matrix $\mathbf{A}$. The update vector $\mathbf{U}$ is computed by the weighted mean for better stability of the attention operation:

$$\mathrm{W}_{i,j} = \frac{\mathrm{A}_{i,j}}{\sum_{l=1}^{N} \mathrm{A}_{l,j}} \tag{2.5}$$

The vector $\mathbf{U}$ is then used to update the slots $\mathcal{S}$ by feeding it to a Gated Recurrent Unit (GRU) [Cho et al., 2014] as input, where slots are the hidden state. The GRU is applied on each slot $\mathcal{S}_i$ separately using shared parameters:

$$\mathcal{S}_i = \mathrm{GRU}(\mathrm{state}{=}\mathcal{S}_i, \mathrm{inputs}{=}\mathbf{U}) \tag{2.6}$$

The slot attention module uses iterative attention, letting the model learn a clustering strategy to decompose input features into a set of slot representations; therefore, this process is repeated multiple times, with each iteration, slots refining themselves to better represent each object. Three iterations are used as default in Locatello et al. [2020]. The slots engage in a competitive process to represent parts of the input scene using the softmax-based attention mechanism [Bahdanau et al., 2014; Luong et al., 2015; Vaswani et al., 2017]. After GRU iterations, the scene is decomposed into slots $\mathcal{S}_i$, each representing an object or background.

The slots are then normalized using Layer Normalization (LN) and passed through a Multi-Layer Perceptron (MLP) with ReLU activation and residual connection [He et al., 2016] to improve the performance of the model.

$$\mathcal{S}_i += \mathrm{MLP}(\mathrm{LN}(\mathcal{S}_i)) \tag{2.7}$$

Like GRU, this residual MLP is applied independently using shared parameters on each slot $\mathcal{S}_i$. The Algorithm 1 summarizes how the slot attention module refines the randomly initialized slots $\mathcal{S}$ through this iterative process.

Each refined slot $\mathcal{S}_i$ is broadcasted to a 2D grid, and position embedding is added. Then a Spatial Broadcast decoder [Watters et al., 2019b] (also referred to as "pixel-mixture decoder" in literature), which comprises multiple deconvolutional layers, is used for upsampling the 2D grid and reconstructing each object and its masks.

Algorithm 1 Slot Attention Module

Input: input features $\mathbf{h}$, randomly initialized slots $\mathcal{S}$

Layer params: q, k, v: linear projections for attention; MLP, GRU; LayerNorm

procedure SLOTATTENTION
  $\mathbf{h} = \text{LayerNorm}(\mathbf{h})$
  for $l = 0 \ldots \text{num\_iterations}$ **do**
    $\mathcal{S}^{\text{prev}} = \mathcal{S}$
    $\mathcal{S} = \text{LayerNorm}(\mathcal{S})$
    $\mathbf{A} = \text{Softmax}(\frac{1}{\sqrt{D}} k(\mathbf{h}) \cdot q(\mathcal{S})$ ▷ Check Eq. 2.2
    $\mathbf{U} = \text{WeightedMean}(\text{weights}=\mathbf{A} + \epsilon, \text{values}=v(\mathbf{h}))$ ▷ Check Eq. 2.4
    for $i = 1 \ldots L$ **do**
      $\mathcal{S}_i = \text{GRU}(\text{state}=\mathcal{S}_i^{prev}, \text{inputs}=\mathbf{U})$
      $\mathcal{S}_i += \text{MLP}(\text{LayerNorm}(\mathcal{S}_i))$
  return $\mathcal{S}$

A softmax function is applied to the masks. The masks are used to compute the weighted sum of the object images, which is the final reconstruction. Mean-squared error is used as the reconstruction loss for training the model.

2.1.2 Slot Attention Transformer

The Slot Attention model from Locatello et al. [2020] works well on synthetic image datasets with simplistic objects like spheres, cubes, and cylinders of uniform color and texture. However, it performs poorly when real-like data is used to train and evaluate this model. Singh et al. [2022a] addressed the shortcomings associated with pixel-mixture decoder and proposed a solution. They indicated that since there is no explicit motivation for the encoder to make a slot focus on the pixel region of the corresponding object, the model relies on the decoder such as Spatial Broadcast Network [Watters et al., 2019b] to make the concept of object emerge in the slots. They argued that the pixel-mixture decoder limits the interaction between the slots and their reconstructions, which causes problems for the slots to be able to

represent complex objects. Hence, they introduced a novel design known as SLATE: Slot Attention Transformer, which employs a powerful autoregressive decoder based on a transformer conditioned on the slots.

SLATE obtains tokens of the input image by using Discrete Variational Autoencoder (dVAE) [Im Im et al., 2017]. The image $\mathbf{x}$ is split into patches $[\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_L]$ for this process.

Each patch $\mathbf{x}_i$ is the input to dVAE encoder f_ϕ^{dVAE}, which outputs the log probabilities $\mathbf{o}_i$ for a categorical distribution:

$$\mathbf{o}_i = f_\phi^{\text{dVAE}}(\mathbf{x}_i) \tag{2.8}$$

Log probabilities $\mathbf{o}_i$ are used to sample relaxed one-hot encoding $\mathbf{z}_i^{\text{soft}}$ for patch i by utilizing relaxed categorical distribution [Jang et al., 2016] with a temperature τ:

$$\mathbf{z}_i^{\text{soft}} \sim \text{RelaxedCategorical}\ (\mathbf{o}_i; \tau) \tag{2.9}$$

The relaxed one-hot encoding $\mathbf{z}_i^{\text{soft}}$ are passed to the dVAE decoder g_θ^{dVAE} and it reconstructs the patch $\tilde{\mathbf{x}}_i$:

$$\tilde{\mathbf{x}}_i = g_\theta^{\text{dVAE}}\left(\mathbf{z}_i^{\text{soft}}\right) \tag{2.10}$$

The MSE reconstruction loss $\mathcal{L}_{\text{dVAE}}$ that is used to train the dVAE encoder f_ϕ^{dVAE} and decoder g_θ^{dVAE}:

$$\mathcal{L}_{\text{dVAE}} = \sum_{i=1}^{T} (\tilde{\mathbf{x}}_i - \mathbf{x}_i) \tag{2.11}$$

Next, the object slots are computed. For this process, in the first step, the log probabilities $\mathbf{o}_i$ are computed using the dVAE encoder f_ϕ^{dVAE} as in Equation 2.8. Then, these log probabilities are used to obtain a discrete token $\mathbf{z}_i$ for each patch i.

$$\mathbf{z}_i \sim \text{Categorical}\,(\mathbf{o}_i) \tag{2.12}$$

Each discrete token $\mathbf{z}_i$ is mapped to an embedding in the next step. The patch embeddings are summed with learned position embeddings $\mathbf{p}_{\phi,i}$ to incorporate the

position information. The resultant embedding $\mathbf{u}_i$ has information about the patch's position as well as its content:

$$\mathbf{u}_i = \text{Dictionary}_\phi\left(\mathbf{z}_i\right) + \mathbf{p}_{\phi,i} \tag{2.13}$$

In the final step, all these patch embeddings $\mathbf{u}_{1:L}$ are passed as input to the Slot Attention encoder [Locatello et al., 2020] as explained in Section 2.1.1. The Slot Attention encoder outputs K slots $\mathcal{S}_{1:K}$ and K attention maps $\mathbf{A}_{1:K}$.

$$\hat{\mathcal{S}}_{1:K}, \mathbf{A}_{1:K} = \text{SlotAttention}\left(\mathbf{u}_{1:L}, \mathcal{S}_{1:K}\right) \tag{2.14}$$

In the last part of the SLATE model, the input image $\mathbf{x}$ is reconstructed by feeding the slots $\hat{\mathcal{S}}_{1:K}$ to an autoregressive transformer-based decoder. This decoding process has multiple steps. In order to predict the log probability of the l^{th} discrete token, slots $\hat{\mathcal{S}}_{1:K}$ and all the previous patch embeddings $\mathbf{u}_{1:l-1}$ are fed to the transformer $g_\theta^{\text{slot-transformer}}$:

$$\hat{\mathbf{o}}_l = g_\theta^{\text{slot-transformer}}\left(\hat{\mathcal{S}}_{1:K}; \mathbf{u}_{1:l-1}\right) \tag{2.15}$$

Lastly, cross-entropy (CE) loss between discrete tokens $\mathbf{z}$ and predicted log probabilities $\hat{\mathbf{o}}$ is calculated:

$$\mathcal{L}_{\text{CE}} = \sum_{l=1}^{L} \text{CE}\left(\mathbf{z}_l, \hat{\mathbf{o}}_l\right) \tag{2.16}$$

The complete model is trained with $\mathcal{L}_{\text{SLATE}}$ which is the sum of the cross-entropy loss $\mathcal{L}_{\text{CE}}$ and the MSE reconstruction loss $\mathcal{L}_{\text{dVAE}}$:

$$\mathcal{L}_{\text{SLATE}} = \mathcal{L}_{\text{CE}} + \mathcal{L}_{\text{dVAE}} \tag{2.17}$$

2.2 Unsupervised Object-Centric Representation Learning in Videos

Object-centric representation learning is not only thriving in the domain of still images, but it also has rich literature for videos. Many studies have approached unsupervised video segmentation and tracking by using recurrent slot-based encoders and trained using reconstructions. Among these studies, some focused on

using bounding boxes for tracking [Kosiorek et al., 2018; He et al., 2019; Stanić and Schmidhuber, 2019; Jiang et al., 2019; Crawford and Pineau, 2020a; Lin et al., 2020a; Wu et al., 2021; Singh et al., 2021]. At the same time, others concentrated on localizing the object by using segmentation masks [Greff et al., 2017; Van Steenkiste et al., 2018; Watters et al., 2019a; Veerapaneni et al., 2020; Du et al., 2020; Creswell et al., 2020; Weis et al., 2021; Kipf et al., 2021; Kabra et al., 2021; Zoran et al., 2021; Besbinar and Frossard, 2021; Creswell et al., 2021; Singh et al., 2022b]. Our research also focuses on the localization of the objects, and the model is trained with the reconstruction loss; however, we also generate the bounding boxes for the tracking performance evaluation. Some other studies explore the utilization of contrastive losses [Kipf et al., 2019; Carvalho et al., 2020]. Another line of work has explored unsupervised object-centric representation learning in 3D scenes [Henderson and Lampert, 2020; Crawford and Pineau, 2020b; Du et al., 2020; Chen et al., 2021; Stelzner et al., 2021; Kabra et al., 2021].

In order to position our research within the context of existing studies and to grasp the foundational notions of our research, we thoroughly examine two specific works: Kipf et al. [2021] and Singh et al. [2022b].

2.2.1 Slot Attention for Videos

In 2021, Kipf et al. introduced the SAVi model for video data, which was based on the slot attention algorithm [Locatello et al., 2020]. They initialized slots of the first frame using weak supervision signals, such as the objects' center point and bounding boxes. They also presented the unsupervised version of their model in which the first frame slots are randomly initialized instead of utilizing weak supervision signals.

An image sequence $\mathbf{x}$ with T frames is input to the model. Here, $\mathbf{x} = [\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots, \mathbf{x}^{(T)}]$. At each frame t, a CNN-based encoder extracts the features $\mathbf{h}^{(t)}$ from the frame $\mathbf{x}^{(t)}$, and position embedding is added to these features. Then, the slots of the first frame are randomly initialized from the Gaussian Distribution with learnable mean and variance:

$$\mathcal{S}^{(0)} \sim \mathcal{N}(\mu, \sigma) \tag{2.18}$$

The extracted features $\mathbf{h}^{(t)}$ and the initialized slots $\mathcal{S}^{(t-1)}$ from the previous frame are fed to the Slot Attention module [Locatello et al., 2020] (as explained in Section 2.1.1). After a number of GRU iterations, the slot attention module outputs the refined slots $\hat{\mathcal{S}}^{(t)}$, that represent the objects and the background in the scene:

$$\hat{\mathcal{S}}^{(t)} = \text{SlotAttention}(\mathbf{h}^{(t)}, \mathcal{S}^{(t-1)}) \tag{2.19}$$

In the next step, SAVi [Kipf et al., 2021] uses a mixture-based decoder similar to the decoder in Locatello et al. [2020]. It broadcasts each slot $\hat{\mathcal{S}}_i^{(t)}$ of each frame to a 2D grid, where $\hat{\mathcal{S}}_i^{(t)}$ represents the i^{th} slot of frame t. Then a Spatial Broadcast decoder [Watters et al., 2019b], g_θ^{mix}, is used for upsampling the 2D grid, which results in reconstructing each object $\mathbf{y}_i^{(t)}$ and its mask $\mathbf{m}_i^{(t)}$:

$$\left(\mathbf{y}_i^{(t)}, \mathbf{m}_i^{(t)}\right) = g_\theta^{\text{mix}}\left(\hat{\mathcal{S}}_i^{(t)}\right) \tag{2.20}$$

To construct the complete final reconstruction of the input image at frame t, each object reconstruction $\mathbf{y}_i^{(t)}$ is multiplied with its masks $\mathbf{m}_i^{(t)}$, and finally, all K slot reconstructions are summed together $\hat{\mathbf{x}}^{(t)}$.

$$\hat{\mathbf{x}}^{(t)} = \sum_{i=1}^{K} \mathbf{y}_i^{(t)} \odot \mathbf{m}_i^{(t)} \tag{2.21}$$

Moreover, this algorithm uses the information from the previous frame's slots to compute the initialization for the next frame's slots $\mathcal{S}^{(t+1)}$. Except for the first frame, the slots are initialized from the previous frame's slots using a module called a predictor. The predictor model uses the Transformer Encoder proposed in Vaswani et al. [2017]:

$$\tilde{\mathcal{S}}^{(t)} = \text{LN}\left(\text{MultiHeadSelfAttn}\left(\hat{\mathcal{S}}^{(t)}\right) + \hat{\mathcal{S}}^{(t)}\right) \tag{2.22}$$

$$\mathcal{S}^{(t)} = \text{LN}\left(\text{MLP}\left(\tilde{\mathcal{S}}^{(t)}\right) + \tilde{\mathcal{S}}^{(t)}\right) \tag{2.23}$$

where, LN is the Layer normalization [Ba et al., 2016], and MLP stands for Multi-Layer Perceptron. This helps to preserve the permutations of the slots, and the slots attend to their corresponding slots from the previous frames. The output of this transformer encoder is projected as the query. $\hat{\mathcal{S}}^{(t)}$ are the refined slots from the current frame, and $\mathcal{S}^{(t)}$ are the slots initialization that will be used in the computation for the next frame like in Equation 2.19.

The mean-squared error between the input frame $\mathbf{x}^{(t)}$ and its reconstruction $\hat{\mathbf{x}}^{(t)}$ is calculated for all the T frames. Lastly, reconstruction loss $\mathcal{L}_{\text{SAVi}}$ is computed by summing the MSE loss of each frame. This loss function is used to train the model:

$$\mathcal{L}_{\text{SAVi}} = \sum_{t=1}^{T} \left\| \mathbf{x}^{(t)} - \hat{\mathbf{x}}^{(t)} \right\|^2 \tag{2.24}$$

As the slot initializations depend on the previous frames, the gradients are back-propagated from the last to the first frame computation. Furthermore, a GRU in the slot attention module is difficult to train on longer sequences. Therefore, the SAVi model is trained on videos with six frames; the training is unstable and often impossible on longer sequences.

2.2.2 Slot Transformer for Videos

Unsupervised SAVi [Kipf et al., 2021] worked well with the datasets with simple objects and backgrounds. However, with real-like datasets, it struggled with learning the object representations. The results were blurry, and the slots encoded similar-looking textures and uniform color regions as the objects. The qualitative and quantitative performance significantly reduced with these datasets. Hence, Singh et al. [2022a] proposed a model that uses an autoregressive slot attention transformer for videos. They named this model STEVE: Slot Transformer for Videos [Singh et al., 2022b].

The STEVE model comprises three modules: a CNN feature extractor, a slot attention encoder that updates the slots for each frame using RNN like SAVi, and lastly, it has a slot transformer decoder. Given an input video $\mathbf{x}$ that has T frames ($\mathbf{x} = [\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots, \mathbf{x}^{(T)}]$) the model aims to decompose the image at each frame

$t \in [1, 2, \dots T]$ into K slots $\hat{\mathcal{S}}^{(t)} = [\hat{\mathcal{S}}_1^{(t)}, \hat{\mathcal{S}}_2^{(t)}, \dots, \hat{\mathcal{S}}_K^{(t)}]$. First, the CNN feature extractor is used to extract the features $\mathbf{h}$ from all input frames $\mathbf{x}$, where, $\mathbf{h} = [\mathbf{h}^{(1)}, \mathbf{h}^{(2)}, \dots, \mathbf{h}^{(T)}]$.

Then, the slots for the first frame are randomly initialized from the Gaussian Distribution with learnable mean and variance:

$$\mathcal{S}^{(0)} \sim \mathcal{N}(\mu, \sigma) \tag{2.25}$$

In the next step, similar to SAVi, the features $\mathbf{h}^{(t)}$ and the initialized slots $\mathcal{S}^{(t-1)}$ from the previous time step are fed to the Slot Attention module [Locatello et al., 2020]:

$$\hat{\mathcal{S}}^{(t)} = \text{SlotAttention}(\mathbf{h}^{(t)}, \mathcal{S}^{(t-1)}) \tag{2.26}$$

After a few iterations, the slot attention module outputs the refined slots $\hat{\mathcal{S}}^{(t)}$ that the slot transformer decoder will use. These slots will also be used by Transformer encoder Vaswani et al. [2017] to initialize the slots for the next frame as described in Equations 2.22 and 2.23.

After the slots for all frames are obtained, the slot transformer decoder reconstructs all input frames $\mathbf{x}$. In this process, each frame $\mathbf{x}^{(t)}$ is treated as a sequence of discrete tokens provided by a discrete VAE encoder f_ϕ^{dVAE}:

$$\mathbf{z}^{(t)} = f_\phi^{\text{dVAE}}\left(\mathbf{x}^{(t)}\right) \tag{2.27}$$

where $\mathbf{z}^{(t)} = [\mathbf{z}_1^{(t)}, \mathbf{z}_2^{(t)}, \dots, \mathbf{z}_L^{(t)}]$ are L discrete tokens that act as prediction targets for the transformer at time t.

The slot-transformer decoder $g_\theta^{\text{slot-transformer}}$ takes all slots $\mathcal{S}_{1:K}^{(t)}$ and all the previous discrete tokens $\mathbf{z}_{1:l-1}^{(t)}$ as input, and learns to predict this sequence of tokens auto-regressively by minimizing the loss $\mathcal{L}_{\text{CE}}$:

$$\hat{\mathbf{o}}_l^{(t)} = g_\theta^{\text{slot-transformer}}\left(\hat{\mathcal{S}}_{1:K}^{(t)}; \mathbf{z}_{1:l-1}^{(t)}\right) \tag{2.28}$$

$$\mathcal{L}_{\text{CE}} = \sum_{t=1}^{T} \sum_{l=1}^{L} \text{CE}\left(\mathbf{z}_l^{(t)}, \hat{\mathbf{o}}_l^{(t)}\right) \tag{2.29}$$

where CE is cross-entropy loss. Like in SLATE, the complete model is jointly trained using this cross-entropy loss $\mathcal{L}_{\text{CE}}$ and MSE reconstruction loss of discrete VAE, $\mathcal{L}_{\text{dVAE}}$, as explained in Equation 2.11:

$$\mathcal{L}_{\text{STEVE}} = \mathcal{L}_{\text{CE}} + \mathcal{L}_{\text{dVAE}} \tag{2.30}$$

Chapter 3

METHODOLOGY

3.1 Memory-Augmented Slot Attention for Videos

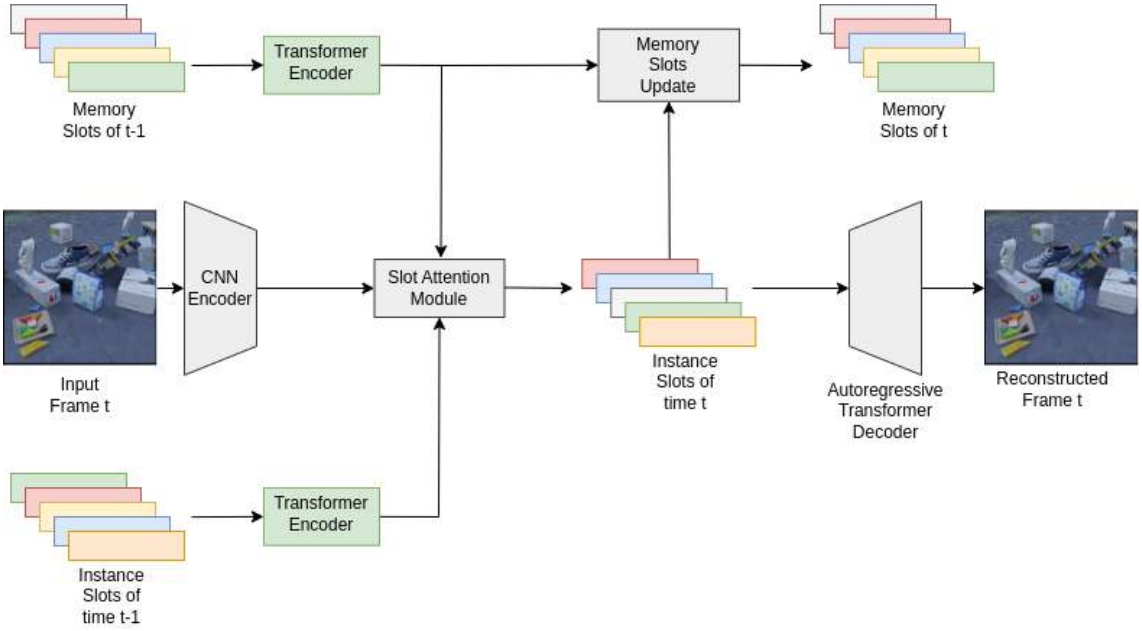


Figure 3.1: Encoder-Decoder Architecture for Memory-Augmented Slot Attention

The prior related works use the information from only the previous frame to compute the slots of the current frame. Thus, they do not rely on long-term dependencies or exploit the videos' temporal information to the full extent. In contrast, we have a basic assumption that reconstructing a scene from the information of the previous frames might aid the object segmentation and tracking. Hence, we created another set of slots, called the memory slots $\mathcal{M}$. The primary purpose of the memory slots is to carry the object information from previous frames to the next. For the slots that decompose each frame, we call them instance slots $\mathcal{S}$. The purpose of the instance slots is the same as in other related works [Locatello et al., 2020; Kipf

et al., 2021; Singh et al., 2022a,b].

Like Kipf et al. [2021]; Singh et al. [2022b], CNN encoder is used to extract the features from the input video $\mathbf{x}$ with T frames. The CNN encoder outputs extracted features $\mathbf{h}$ for all frames; $\mathbf{h} = [\mathbf{h}^{(1)}, \mathbf{h}^{(2)}, \ldots, \mathbf{h}^{(T)}]$. A positional embedding is added to these features and fed to our memory-conditioned slot attention module. In our architecture, we have two transformer encoders to initiate instance and memory slots from the previous frames.

3.1.1 Slot Attention with Memory

The memory-conditioned slot attention module takes CNN features, instance slots, and memory slots as input and returns the refined slot for the current frame. For the first frame, the instance slots $\mathcal{S}^{(0)}$ are randomly initialized using Gaussian distribution:

$$\mathcal{S}^{(0)} \sim \mathcal{N}(\mu, \sigma) \tag{3.1}$$

The slot attention encoder from Locatello et al. [2020] is used to refine the slots. It uses instance slots $\mathcal{S}^{(0)}$ as query, and CNN features of the first frame $\mathbf{h}^{(1)}$ as key and value. The slot attention encoder outputs the refined slots for the first frame $\mathcal{S}^{(1)}$, and that is passed to the autoregressive transformer-based decoder as in Singh et al. [2022b].

The memory slots play a role in the slot attention computations after the first frame slot attention encoder. Since there is no memory for the first frame, its computations are done without memory slots. The first memory slots $\mathcal{M}^{(1)}$ are initialized by cloning the refined instance slots $\mathcal{S}^{(1)}$ of the first frame. For the second and consequent frames, instance slots and memory slots are initialized by using different transformer encoders with take respective previous instance slots and memory slots as inputs:

$$\mathcal{S}^{(t-1)} = \text{TransformerEncoder}(\mathcal{S}^{(t-1)}) \tag{3.2}$$

$$\mathcal{M}^{(t-1)} = \text{TransformerEncoder}(\mathcal{M}^{(t-1)}) \tag{3.3}$$

Linear projection is applied to the instance slots $\mathcal{S}^{(t)}$, and the result serves as the query for the slot attention mechanism. Two other linear transformations project input features as key and value. Layer normalization [Ba et al., 2016] is used after each projection. Using the slot attention mechanism, the input $\mathbf{U}_1^{(t)}$ for the GRU is computed. Another vector is computed by using memory slots. For the second and subsequent frames, for each GRU iteration, we use a transformer decoder [Vaswani et al., 2017] to perform cross-attention between the instance and memory slots, which outputs $\mathbf{U}_2^{(t)}$.

The transformer decoder first applies multi-head self-attention on instance slots, adds a residual connection, and performs layer normalization [Ba et al., 2016]:

$$\tilde{\mathcal{S}}^{(t-1)} = \text{LN}\left(\text{MultiHeadSelfAttn}\left(\mathcal{S}^{(t-1)}\right) + \mathcal{S}^{(t-1)}\right) \tag{3.4}$$

Then multi-head cross attention is applied to the temporary variable $\tilde{\mathcal{S}}^{(t-1)}$ from Equation 3.4 and the memory slots from previous frame $\mathcal{M}^{(t-1)}$. Residual connection is added, and layer normalization is applied:

$$\bar{\mathcal{S}}^{(t-1)} = \text{LN}\left(\text{MultiHeadCrossAttn}\left(\tilde{\mathcal{S}}^{(t-1)}, \mathcal{M}^{(t-1)}\right) + \tilde{\mathcal{S}}^{(t-1)}\right) \tag{3.5}$$

In the last step of the transformer decoder, the temporary variable $\bar{\mathcal{S}}^{(t-1)}$ from Equation 3.5 is passed through a multi-layer perceptron, residual is added, and layer normalization is applied. At this stage, the instance slots are transformed before they are used for the attention computation:

$$\mathbf{U}_1^{(t)} = \text{LN}\left(\text{MLP}\left(\bar{\mathcal{S}}^{(t-1)}\right) + \bar{\mathcal{S}}^{(t-1)}\right) \tag{3.6}$$

The final input for the GRU update is computed by summing $\mathbf{U}_1^{(t)}$ and $\mathbf{U}_2^{(t)}$ and previous instance slots are state of the GRU. The output of GRU is passed through another MLP with an additional residual connection, resulting in updated instance slots. In each GRU iteration, the latest updated instance slots and the memory slots from the previous frame become the input of the transformer decoder, which outputs the query of that iteration. In this manner, the query is initialized for each iteration in each frame. This procedure is visualized in Figure 3.2, and the pseudocode is given in Algorithm 2.

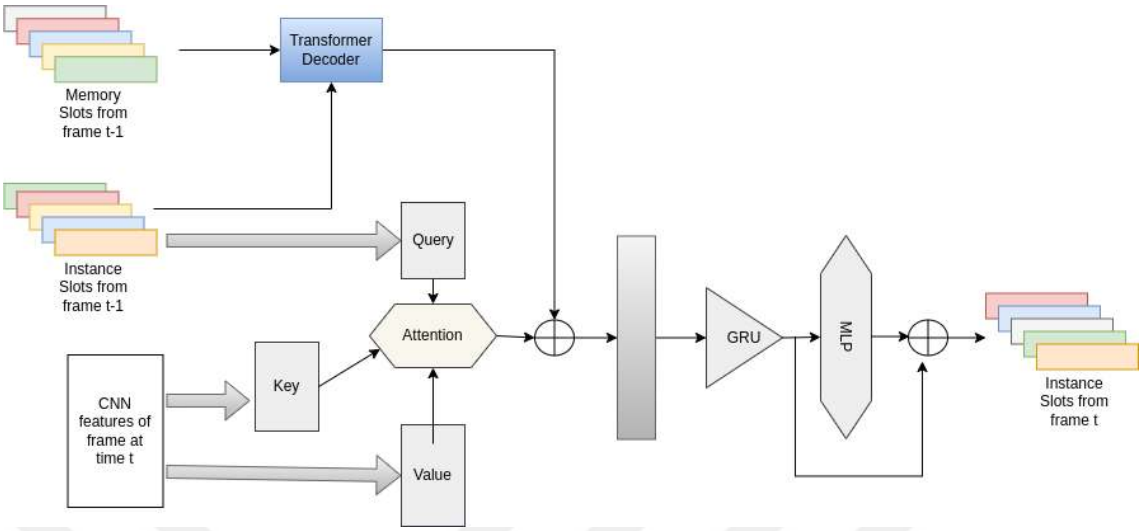


Figure 3.2: Memory-Augmented Slot Attention Module

The slots from the current frame $\mathcal{S}^{(t)}$ are given to the autoregressive transformer decoder Singh et al. [2022b] which reconstructs the input frame. Figure 3.1 shows the end-to-end model architecture.

3.1.2 Updating the Memory Slots

The memory slots are updated at each frame t using the memory slots from the previous frame $\mathcal{M}^{(t-1)}$ and instance slots from the current frame $\mathcal{S}^{(t)}$. Another transformer decoder [Vaswani et al., 2017] is used to update the memory slots. However, here, the memory slots are used as queries, and instance slots are used as keys and values.

First, multi-head self-attention is applied on memory slots, a residual connection is added, and layer normalization is performed:

$$\tilde{\mathcal{M}}^{(t-1)} = \text{LN}\left(\text{MultiHeadSelfAttn}\left(\mathcal{M}^{(t-1)}\right) + \mathcal{M}^{(t-1)}\right) \tag{3.7}$$

Then multi-head cross attention is computed between the temporary variable $\tilde{\mathcal{M}}^{(t-1)}$ from Equation 3.7 and the instance slots from current frame $\mathcal{S}^{(t)}$. Residual connection is added, and layer normalization is applied:

$$\bar{\mathcal{M}}^{(t-1)} = \text{LN}\left(\text{MultiHeadCrossAttn}\left(\tilde{\mathcal{M}}^{(t-1)}, \mathcal{S}^{(t)}\right) + \tilde{\mathcal{M}}^{(t-1)}\right) \tag{3.8}$$

Algorithm 2 Memory-Augmented Slot Attention Module

Input: input features from current frame $\mathbf{h}^{(t)}$, instance slots from previous frame $\mathcal{S}^{(t-1)}$, memory slots from previous frame $\mathcal{M}^{(t-1)}$

Layer params: q, k, v: linear projections for attention; MLP, GRU; LayerNorm; TransformerDecoder

procedure SLOTATTENTION
  $\mathbf{h}^{(t)} = \text{LayerNorm}(\mathbf{h}^{(t)})$
  for $l = 0 \dots \text{num\_iterations}$ **do**
    $\mathcal{S}^{\text{prev}} = \mathcal{S}^{(t-1)}$
    $\mathbf{A}^{(t)} = \text{Softmax}(\frac{1}{\sqrt{D}} k(\mathbf{h}) \cdot q(\mathcal{S}^{(t-1)})$ ▷ Check Eq. 2.2
    $\mathbf{U}_1^{(t)} = \text{WeightedMean}(\text{weights}=\mathbf{A}^{(t)} + \epsilon, \text{values}=v(\mathbf{h}^{(t)}))$ ▷ Check Eq. 2.4
    $\mathbf{U}_2^{(t)} = \text{TransformerDecoder}(\mathcal{S}^{(t-1)}, \mathcal{M}^{(t-1)})$
    $\mathbf{U}^{(t)} = \mathbf{U}_1^{(t)} + \mathbf{U}_2^{(t)}$
    for $i = 1 \dots L$ **do**
      $\mathcal{S}_i^{(t-1)} = \text{GRU}(\text{state}=\mathcal{S}_i^{\text{prev}}, \text{inputs}=\mathbf{U}^{(t)})$
      $\mathcal{S}_i^{(t-1)} += \text{MLP}(\text{LayerNorm}(\mathcal{S}_i^{(t-1)}))$
  $\mathcal{S}^{(t)} = \mathcal{S}^{(t-1)}$
  return $\mathcal{S}^{(t)}$

Finally, the temporary variable $\bar{\mathcal{S}}^{(t-1)}$ from Equation 3.8 is passed through an MLP, residual is added, and layer normalization is applied:

$$\hat{\mathcal{M}}^{(t-1)} = \text{LN}\left(\text{MLP}\left(\bar{\mathcal{M}}^{(t-1)}\right) + \bar{\mathcal{M}}^{(t-1)}\right) \tag{3.9}$$

Before the output of the transformer decoder is passed through the GRU, a residual connection is applied:

$$\hat{\mathcal{M}}^{(t-1)} = \hat{\mathcal{M}}^{(t-1)} + \mathcal{M}^{(t-1)} \tag{3.10}$$

At the end, $\hat{\mathcal{M}}^{(t-1)}$ is passed as the input through the GRU and MLP with a residual connection, and in this manner, memory slots are updated. Unlike instance slots, memory slots are updated only once per frame. It is shown in Figure 3.3

how the memory slots are updated using instance slots, and the pseudocode of this procedure is described in Algorithm 3.

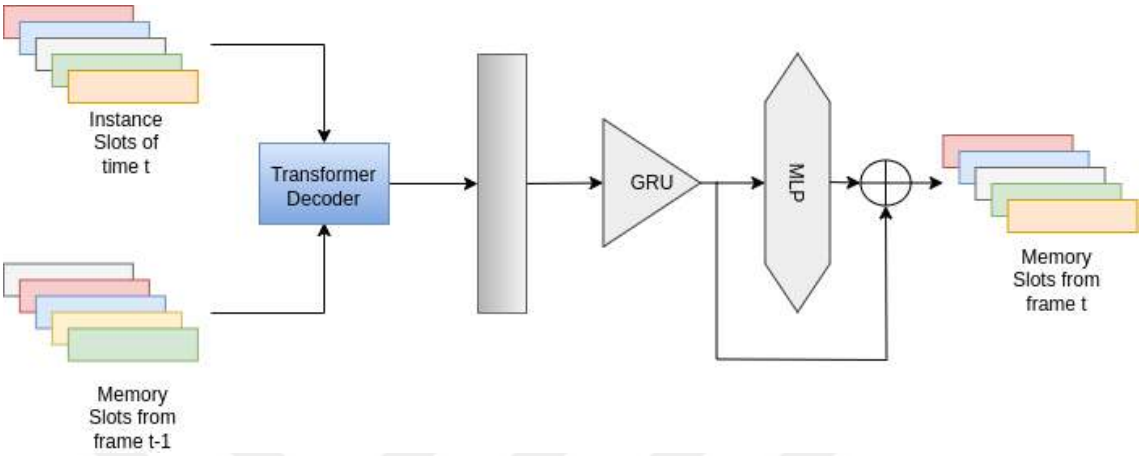


Figure 3.3: Memory Slots Update Module

Algorithm 3 Memory Slots Update Module

Input: instance slots from the current frame $\mathcal{S}^{(t)}$, memory slots from the previous frame $\mathcal{M}^{(t-1)}$

Layer params: GRU; TransformerDecoder

procedure UPDATEMEMORYSLOTS
  $\mathcal{M}^{\text{prev}} = \mathcal{M}^{(t-1)}$
  $\mathcal{M}^{(t-1)} = \text{TransformerDecoder}(\mathcal{M}^{(t-1)}, \mathcal{S}^{(t)})$
  $\mathcal{M}^{(t-1)} = \text{GRU}(\text{state}=\mathcal{M}^{\text{prev}}, \text{inputs}=\mathcal{M}^{(t-1)})$
  $\mathcal{M}^{(t)} += \text{MLP}(\text{LayerNorm}(\mathcal{M}^{(t-1)}))$
  return $\mathcal{M}^{(t)}$

3.2 Training Strategies

We have employed various training methodologies to attain optimized and stable training outcomes. To ensure the stability of the training process for Gated Recurrent Units (GRUs) across extended sequences, the application of truncated backpropagation through time (TBPTT) has been pivotal.

3.2.1 Truncated Backpropagation Through Time

Our contribution builds upon the current state-of-the-art by introducing memory slots that help carry information from previous frames to subsequent ones. To effectively train these memory slots to propagate information across longer sequences, it becomes crucial to employ long sequences during training. However, our model incorporates multiple GRUs, which imposes limitations on the length of sequences we can use. Attempting to work with sequences exceeding more than ten frames leads to unstable training and generally yields unsatisfactory outcomes. To address this challenge, we have integrated a solution known as Truncated Backpropagation Through Time (TBPTT) [Williams and Peng, 1990][Elman, 1990]. In this solution, the input batch is partitioned along the temporal dimension into segments known as splits. Each split represents a window within the video, in our case, encompassing a span of 6 frames. These splits are sequentially supplied to the network, where the associated loss is computed and subsequently utilized for gradient backpropagation throughout the network. This strategic approach alleviates the necessity for the recurrent neural network (RNN) to engage in backpropagation over longer sequences; rather, the process is confined to a limited number of frames within each split. This strategic training method substantially improved object recognition and object tracking aspects of our model's performance. The pseudocode for truncated backpropagation through time for our model is given in Algorithm 4.

Algorithm 4 Truncated Backpropagation Through Time (TBPTT)

Input: Sequence of input data $\mathbf{x}$, Sequence length T, Truncated length P

procedure TRAININGITERATION
    Randomly initialize GRU hidden state $\mathcal{S}^{(0)} \sim \mathcal{N}(\mu, \sigma)$
    for $t = 1$ to T **do**
        $\mathcal{S}^{(t)} = \text{SlotAttention}(\mathbf{x}^{(t)}, \mathcal{S}^{(t-1)})$ ▷ Update hidden state
        if $((t \mod P) == 0)$ or $(t == T)$ **then**
            loss = compute_loss($\mathcal{S}$)
            gradients = compute_gradients(loss)
            update_model(gradients)

Chapter 4

EXPERIMENTAL EVALUATION

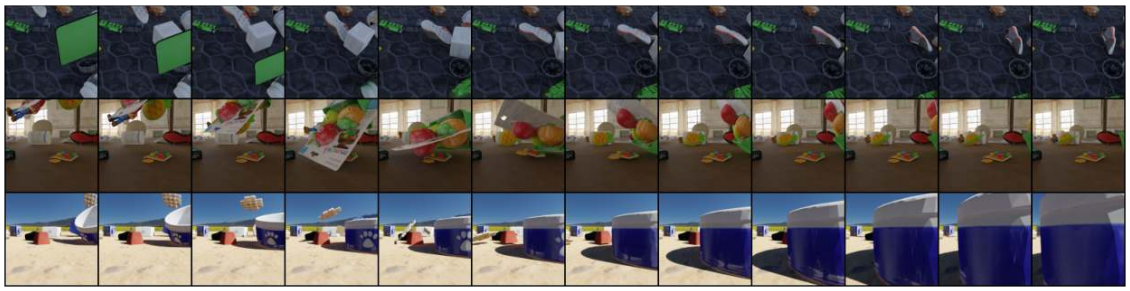
4.1 Datasets

In our research, we used two primary datasets, MOVI-D and MOVI-E, as described in the work by Greff et al. [2022]. These datasets were chosen due to their realistic depiction of objects in motion within dynamic scenes.

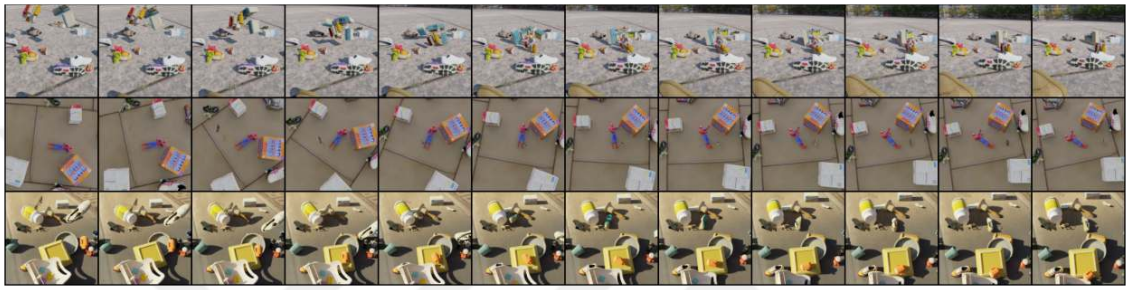
Each video in these datasets comprises 24 frames. There are 9750 training videos and an additional 250 videos for validation. Each frame in these videos has a resolution of 128×128 pixels.

It is important to note that these datasets come with various annotations, but for our evaluation, we exclusively utilized the segmentation masks.

One key distinction between MOVI-D and MOVI-E is the nature of the camera setup. In MOVI-D, the camera remains stationary throughout the videos, while in MOVI-E, the camera exhibits simple linear camera movement. In other words, the background in MOVI-E is not stationary. This difference in camera behavior adds an additional layer of complexity to MOVI-E, making it a more challenging dataset for evaluation. The sample frames from the video sequences of MOVI-D and MOVI-E are shown in Figure 4.1.



(a)



(b)

Figure 4.1: Samples frames from (a) MOVI-D and (b) MOVI-E

4.2 Evaluation Metrics

To analyze our results quantitatively, we assess the performance of our model in terms of both object segmentation and multi-object tracking.

4.2.1 Object Segmentation

We evaluate the performance of object segmentation in each frame of the video using the Adjusted Rand Index (ARI) metric [Rand, 1971]. This ARI is a metric to measure how well the predicted segmentation masks match with the actual ground-truth masks. The metric is invariant for the different permutations of the segmentation masks. This feature makes it particularly fitting for methods that do not rely on manual annotations or supervision. Similar to previous studies [Greff et al., 2019; Locatello et al., 2020; Kipf et al., 2021; Kabra et al., 2021; Singh et al., 2022b], we focus our ARI assessment solely on the foreground objects, which is referred to as Foreground-ARI or FG-ARI. This enables us to measure how effectively our method

distinguishes the objects within the video sequence.

4.2.2 Multi-Object Tracking

In evaluating Multi-Object Tracking (MOT) performance, we use four different metrics each of which provides valuable insights into the effectiveness of the tracking system [Bernardin and Stiefelhagen, 2008].

The Multiple Object Tracking Accuracy (MOTA) metric comprehensively evaluates tracking accuracy by considering false positives FP, false negatives FN, and identity switches $IDSW$. It provides a comprehensive measure of the overall tracking performance.

$$\text{MOTA} = 1 - \frac{\sum_{t=1}^{T}(FN^{(t)} + FP^{(t)} + IDSW^{(t)})}{\sum_{t=1}^{T} GT^{(t)}} \tag{4.1}$$

In addition to MOTA, metrics like Mostly Tracked (MT) and Mostly Lost (ML), are also informative. Mostly Tracked signifies the proportion of objects tracked for at least 80% of their existence in the video sequence. For a high MT score, it is important for the object IDs to remain the same throughout the sequence. Mostly Lost, on the other hand, represents the percentage of time instances in which objects are lost for 80% of their existence in the video sequence. These metrics provide insights into the overall consistency of tracking throughout the sequence.

These evaluation metrics collectively offer a comprehensive understanding of the performance of Multi-Object Tracking methods in terms of accuracy and continuity. Their utilization enables a thorough assessment of the tracking algorithms' capabilities. To keep our implementation and results consistent with other works, we used the guidelines for the metrics from [Milan et al., 2016].

4.3 Experimental Setup

For training with truncated back-propagation, the full-length videos are split into windows of 6 consecutive frames. The splits are fed to the network sequentially. The gradients are computed and backpropagated after each window. The algorithm

is explained in Section 3.2.1. We split the training for training and validation with a ratio of 80:20, and the validation set is used for testing the algorithms. We train all the models for 100 epochs, amounting to approximately 200k iterations. We use two iterations of Slot Attention per frame. Adam optimizer [Kingma and Ba, 2014] is used. On Tesla V100 GPU with 32GB, training our model with truncated backpropagation through time takes about 72 hours for videos with 128 $\times$ 128 resolution. Further architecture details and hyperparameters are provided in Section 4.4.

4.4 Hyperparameters and Architecture Details

We use the default hyperparameters from Singh et al. [2022b] and tune them according to our memory constraints and stable training. The hyperparameters are given in Table 4.1.

Table 4.1: Hyperparameters

Hyperparameter	**Value**
Learning rate (DVAE)	0.0003
Learning rate (Encoder)	0.0001
Learning rate (Decoder)	0.0003
Learning rate warm-up steps	300000
Learning rate half-life steps	300000
Gradient Clip	0.05
Epochs	100
Batch size	4
Image resolution	128×128
Slot size	192
Number of instance slots	15
Number of memory slots	15
Number of GRU iterations	2

Our model comprises a memory-augmentation slot attention encoder and an autoregressive transformer decoder. The parameters of our models were configured to be consistent with the model parameters of Kipf et al. [2021] and Singh et al. [2022b] for fair comparison. The parameters of the CNN encoder and memory-conditioned slot attention module are given in Table 4.2, and the parameters of the decoder are given in Table 4.3.

Table 4.2: Slot Encoder Parameters.

Parameter	**Size/Channels**
CNN hidden size	64
CNN layers	4
MLP hidden size	192
Transformer decoder blocks (Slot Attention)	1
Transformer decoder attention heads (Slot Attention)	4
Transformer decoder blocks (Memory Slots Update)	1
Transformer decoder attention heads (Memory Slots Update)	4

Table 4.3: Autoregressive Transformer Decoder Parameters.

Parameter	**Size/Channels**
Vocabulary size	4096
τ_{start}	1.0
τ_{end}	0.1
τ steps	30000
Transformer decoder blocks	8
Transformer decoder attention heads	4

4.5 Experimental Results

In this section, we examine the performance of our model in comparison with the two baselines which are state-of-the-art in unsupervised object-centric scene decom-

position. We use the unsupervised version of SAVi [Kipf et al., 2021] and STEVE [Singh et al., 2022b] as baseline models. Comparison with STEVE is important because our model is most similar to it with one main difference: our model uses a memory-augmented slot attention encoder while STEVE uses the conventional slot attention encoder. Thus, the comparison analysis of our model with STEVE provides a clear insight into our contribution. We keep the model parameters, number of slots, and slot size consistent for all the models for fair comparison.

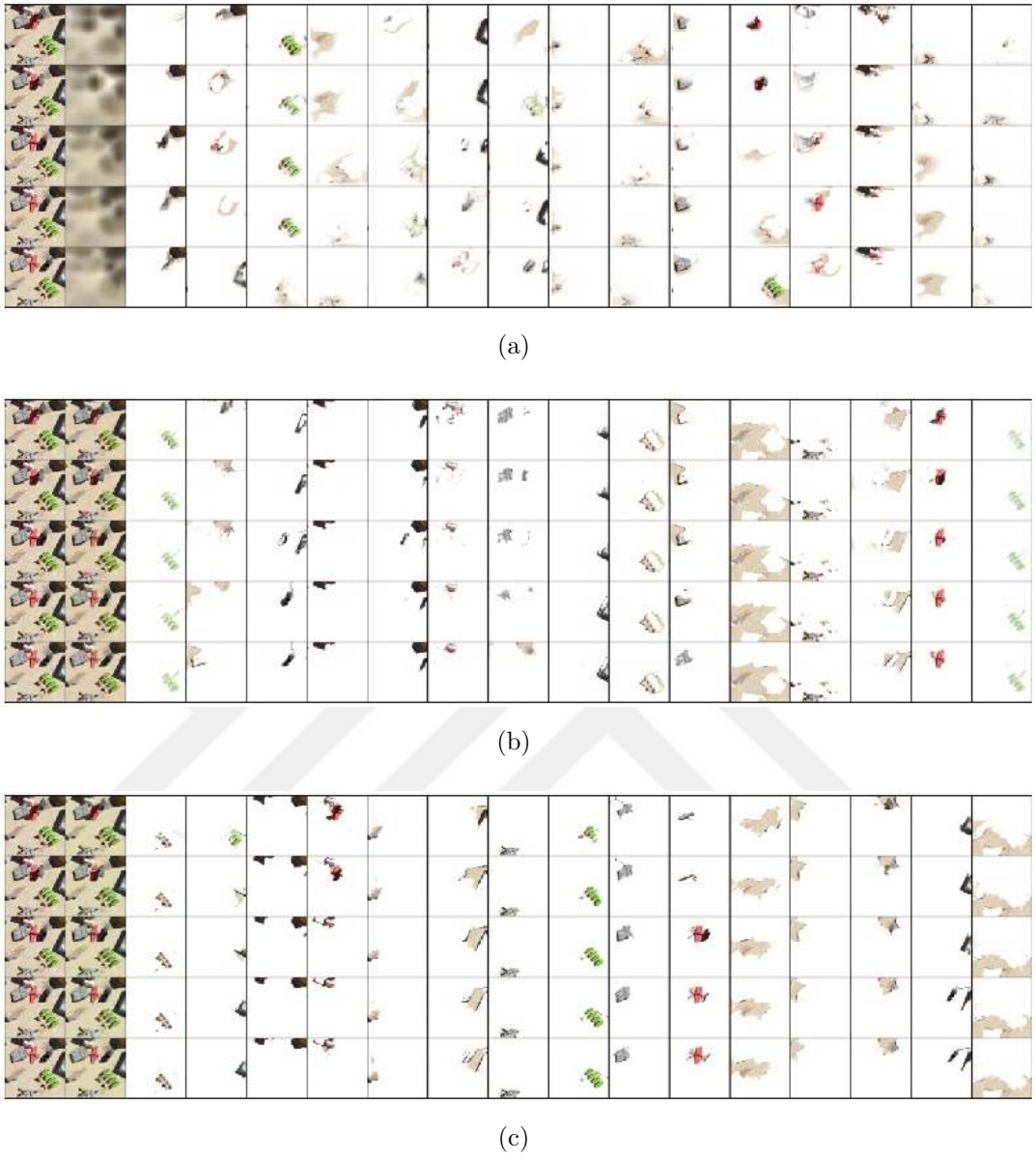
4.5.1 Object Segmentation

Table 4.4 presents the FG-ARI (Foreground Adjusted Rand Index) scores for different models trained using TBPTT across various numbers of frames for evaluation. We also evaluated the models on 48 frames with video augmentation by mirroring the sequence. For the mirrored sequence, we flip the input video and concatenate it to the input video. In the evaluation of different models on the MOVI-D and MOVI-E datasets, we observe that our model consistently achieved the highest FG-ARI scores across varying numbers of frames for evaluation. This indicates that our model excels in accurately representing and segmenting objects in complex video scenes compared to other models, demonstrating its effectiveness in object-centric representation learning. The good performance of our model for 24 and 48 frames in both MOVI-D and MOVI-E datasets further demonstrates its proficiency in capturing temporal dynamics and object-centric representations in video data.

Table 4.4: FG-ARI Score of Models trained using TBPTT.

Dataset	**Model**	**Number of Frames for Evaluation**				
		6	12	18	24	48
MOVI-D	SAVi	31.96	28.79	24.36	23.46	17.86
	STEVE	48.54	48.42	47.60	47.41	46.06
	Our Model	50.20	49.97	49.08	48.84	48.10
MOVI-E	SAVi	31.09	29.24	27.43	26.79	20.66
	STEVE	50.87	51.94	51.88	52.03	51.79
	Our Model	51.60	54.35	53.13	53.07	52.40

Moreover, our model performs well in terms of qualitative results. The visual results are shown in Figures 4.2 and 4.3. It can be seen that the results are much better than SAVi and similar to that of STEVE. However, when we observe closely, STEVE splits the objects more than our model. In comparison, our model splits the background more than STEVE. Thus resulting in a slightly lower FG-ARI score.

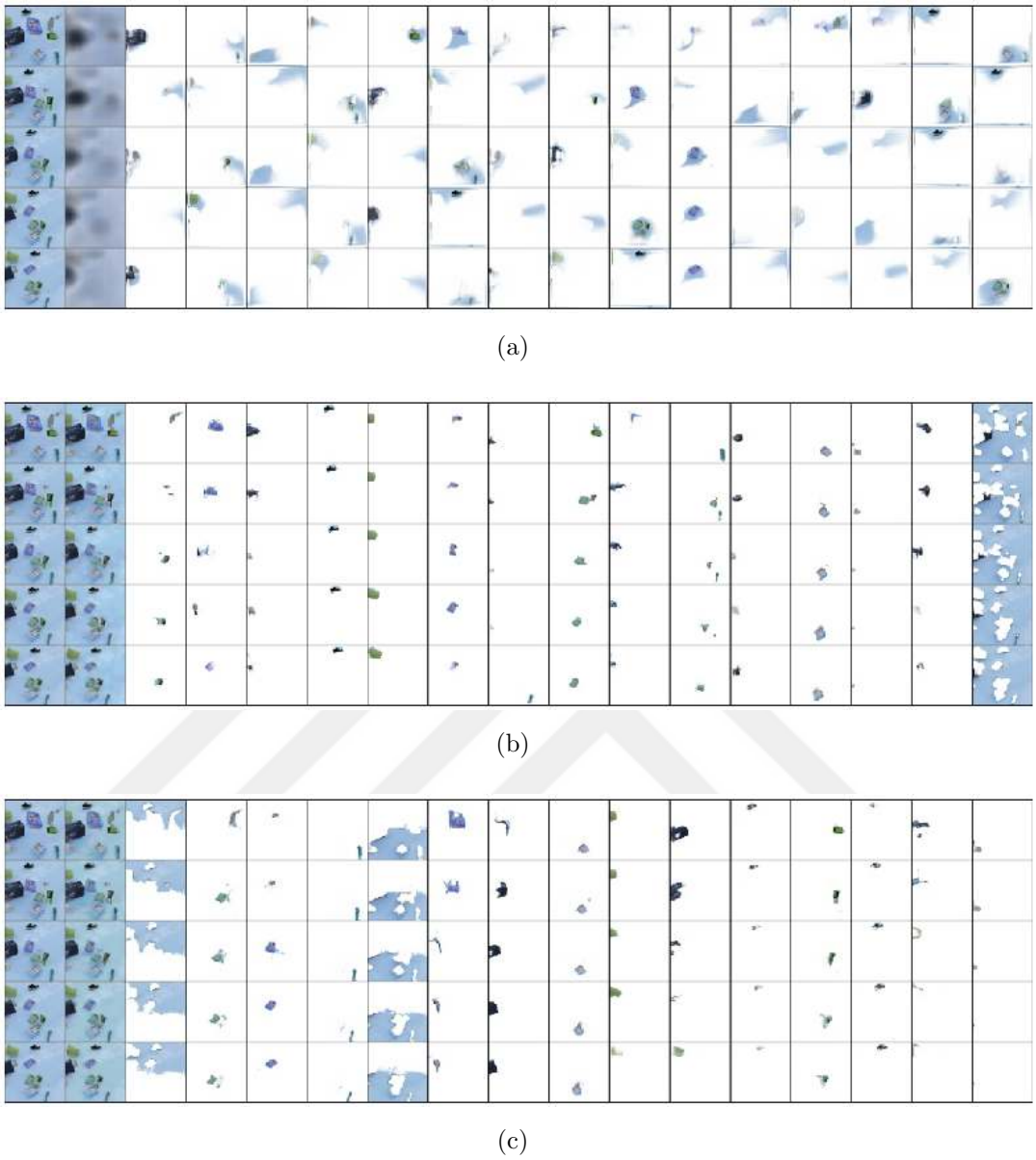


(a)

(b)

(c)

Figure 4.2: Frame-wise Object Segmentation Results from MOVI-D of (a) SAVi, (b) STEVE, and (c) Our Model. The rows are frames sampled at 1 sample per 4 frames. The first column is the input frames, the second column is the full reconstruction and the following columns are the slot reconstructions.



(a)

(b)

(c)

Figure 4.3: Frame-wise Object Segmentation Results from MOVI-E of (a) SAVi, (b) STEVE, and (c) Our Model. The rows are frames sampled at 1 sample per 4 frames. The first column is the input frames, the second column is the full reconstruction, and the following columns are the slot reconstructions.

4.5.2 Multi-Object Tracking

We analyze the multi-object tracking performance on two datasets, MOVI-D and MOVI-E, by evaluating over a full sequence of 24 frames. Table 4.5 provides a detailed analysis of our results. For the MOVI-D dataset, we observe that our model achieved the highest MOTA among the models, indicating its accuracy in

tracking multiple objects over time. Additionally, our model outperforms both SAVi and STEVE in MT and ML, demonstrating its precision and reliability in tracking. Similarly, for the MOVI-E dataset, our model maintains the highest performance compared to other models in terms of all tracking metrics. These results suggest our model excels in multi-object tracking tasks, delivering superior accuracy and precision on both datasets.

The tracking performance can be noticed in Figures 4.2 and 4.3. Moreover, the full sequence visual results of our model and the baseline models are given in Appendix A in Figures: A.1, A.2, and A.3. It can be easily noticed that our model is able to track objects without switching the identities of the objects. Since the objects are less prone to split into two or more slots, the MOTA score is high, and it aids the tracking process. For our model, it can be seen that if an object exits the scene, its corresponding slot remains empty.

Table 4.5: Multi-Object Tracking Evaluation on Full Sequence (24 frames).

Dataset	**Model**	**MOTA**	**MT**	**ML**
MOVI-D	SAVi	-74.3	8.42	64.22
	STEVE	-34.7	38.3	42.14
	Our Model	-13.8	48.32	31.89
MOVI-E	SAVi	-61.1	7.05	71.58
	STEVE	-6.1	40.41	37.51
	Our Model	4.9	46.05	32.17

We extend the multi-object tracking evaluation over longer sequences. For this purpose, we use two video augmentations: "repeated sequence" and "mirrored sequence," each consisting of 48 frames. For the repeated sequence, we repeat the video twice, and for the mirrored sequence, we flip the input video and concatenate it to the input video. Consequently, in the repeated sequence, there will be a sudden jump for all the objects, whereas the object motion in the mirrored sequence is smoother compared to the repeated sequence.

Quantitative results for the repeated sequence are given in Table 4.6, and for mirrored sequences are given in Table 4.7. In both sequences, our model consistently outperforms the SAVi and STEVE models in terms of MOTA, indicating its high accuracy in tracking objects. It also exhibits a high MT and low ML, signifying better tracking.

Overall, these results underscore the robust multi-object tracking performance of our model, characterized by high accuracy, precision of the detections, and tracking across both the repeated and mirrored sequences. While our model outperforms all other models for all the settings in multi-object tracking metrics, the negative MOTA, low MT, and high ML values suggest that there is still room for improvement.

Table 4.6: Multi-Object Tracking Evaluation on Repeated Sequence (48 frames).

Dataset	**Model**	**MOTA**	**MT**	**ML**
MOVI-D	SAVi	-75.1	8.82	64.89
	STEVE	-33.8	38.68	41.22
	Our Model	-13.5	48.73	31.17
MOVI-E	SAVi	-61.0	7.16	72.20
	STEVE	-4.7	40.65	36.73
	Our Model	5.6	46.42	32.14

Table 4.7: Multi-Object Tracking Evaluation on Mirrored Sequence (48 frames).

Dataset	**Model**	**MOTA**	**MT**	**ML**
MOVI-D	SAVi	-75.2	8.83	64.78
	STEVE	-33.3	39.12	49.39
	Our Model	-11.7	49.83	31.65
MOVI-E	SAVi	-61.1	7.16	72.67
	STEVE	-3.1	42.42	36.70
	Our Model	7.6	48.20	32.14

4.6 Ablation Studies

In this section, the contribution of different components of our model is discussed. In particular, we discuss the impact of training the models with truncated backpropagation through time and using an autoregressive transformer decoder for object reconstructions.

4.6.1 Effect of Truncated Backpropagation Through Time

To train our model more effectively, we use truncated backpropagation through time. We access the contribution of TBPTT for our model and other baselines. For the experiment setup, we split the full-length video with 24 frames into consecutive subsequences of 6 frames each. For training without TBPTT, we train the model with each of these 6-frame windows as independent training samples, and for training with TBPTT, the windows are sequentially fed to the network, and after every window, the backpropagation is performed. The algorithm is explained in detail in Section 3.2.1.

We evaluate the performance of the models on the MOVI-D dataset. The number of frames for evaluation ranges from 6 to 24 in order to account for the performance on short and long sequences. The FG-ARI scores with and without TBPTT are given in Table 4.8. It can be seen in the table that the performance of STEVE and our model significantly improves by using TBPTT for both short and long sequences. However, SAVi does not exhibit any improvement; instead, the performance deteriorated slightly by using TBPTT for training.

Table 4.8: Comparison of FG-ARI Scores for models trained with and without Truncated Backpropagation Through Time (TBPTT) on MOVI-D Dataset.

Model	**TBPTT**	**Number of Frames for Evaluation**			
		6	12	18	24
SAVi	✗	34.79	29.59	27.21	24.91
	✓	31.96	28.79	24.36	23.46
STEVE	✗	44.76	44.97	44.01	43.79
	✓	48.54	48.42	47.60	47.41
Our Model	✗	47.50	47.71	45.87	46.08
	✓	50.20	49.97	49.08	48.84

We extend the assessment of the contribution of TBPTT to object tracking. Table 4.9 shows tracking metrics for models trained with and without TBPTT. We evaluate the performance on full sequences of 24 frames from the MOVI-D dataset. The results are consistent with the object segmentation results. The performance of STEVE and our model improves greatly with TBPTT, and it drops for SAVi. It is worth noting that the Most Tracked (MT) and Mostly Lost (ML) results improved more than 2.7 times for our model by using TBPTT.

Table 4.9: Comparison of tracking metrics for different models with and without TBPTT on full sequences (24 frames) from MOVI-D dataset.

Model	**TBPTT**	**MOTA**	**MT**	**ML**
SAVI	✗	-72.1	17.06	56.62
	✓	-74.3	8.42	64.22
STEVE	✗	-48.3	29.74	48.47
	✓	-34.7	38.3	42.14
Our Model	✗	-39.5	35.05	43.76
	✓	-13.8	48.32	31.89

4.6.2 Autoregressive Transformer Decoder vs Mixture-based Decoder

The type of decoder used in the architecture has a great impact on the qualitative and quantitative results. We evaluate the contribution of the autoregressive transformer decoder in comparison with the mixture-based decoder. In these experiments, we changed our autoregressive transformer decoder with a mixture-based decoder. We evaluate the mixture-based decoder with two different settings: trained from scratch and trained using transfer learning. For the mixture-based decoder with transfer learning, we take the slots from the pre-trained encoder of our model, and without propagating the gradient to the encoder, we train only the decoder to reconstruct the objects from the slots. The FG-ARI scores of our model with different decoders are shown in Table 4.10. These results are from models trained and evaluated using the MOVI-D dataset with 24 frames in each video. All these models are trained using TBPTT. We can see that the FG-ARI score of our model with an autoregressive transformer decoder is more than 1.5 times that of the mixture-based decoder trained from scratch.

Table 4.10: Object segmentation performance of our model with different decoders on MOVI-D (24 frames).

Decoder	**FG-ARI**
Mixture-Based Decoder	30.78
Mixture-Based Decoder (using transfer learning)	48.84
Autoregressive Transformer Decoder	49.04

The quality of the visual results varies significantly among models with different decoders. The slot decomposition is shown in Figure 4.4. The outputs generated by mixture-based decoders appear blurry and resemble shapeless blobs. The model with a mixture-based decoder trained using transfer learning performs better than the model trained without pretraining. This indicates decoders play an important role in slot formation. The objects reconstructed from the autoregressive transformer decoder are sharp, and the segmentation results are visible better than others. Despite

the imperfect reconstructions, the object-tracking outcomes remain satisfactory for all the models.

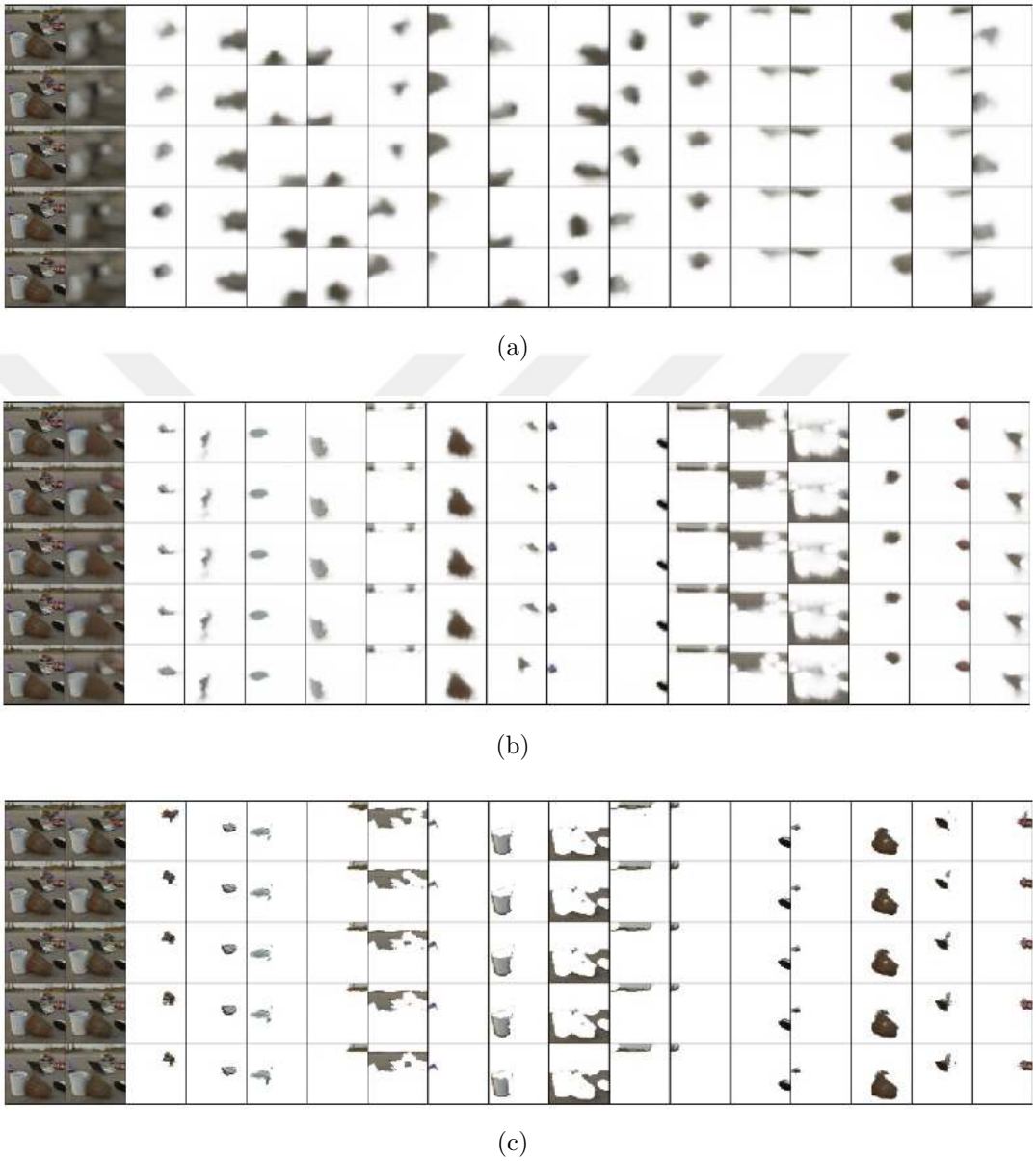


Figure 4.4: Our model qualitative results with different decoders: (a) Mixure-Based Decoder, (b) Mixture-Based Decoder (using transfer learning), and (c) Autoregressive Transformer Decoder. The rows are frames sampled at 1 sample per 4 frames. The first column is the input frames, the second column is the full reconstruction, and the following columns are the slot reconstructions.

We also assess the tracking performance of our model with different decoders. The quantitative results of MOT metrics are shown in Table 4.11. It can be seen that

the mixture-based decoder model trained from scratch performs poorly in terms of MOT metrics. The MOTA score for the mixture-based decoder is -76.1, indicating a high number of false positives. However, when this model is trained using transfer learning, the performance improves more than the autoregressive transformer decider. The differences in MT and ML scores are significantly greater among different decoders. In summary, the Autoregressive Transformer Decoder and mixture-based model, which is trained using transfer learning, significantly outperform the decoder in terms of overall tracking accuracy.

Table 4.11: Tracking performance of our model with different decoders on MOVI-D (24 frames).

Decoder	**MOTA**	**MT**	**ML**
Mixture-Based Decoder	-76.1	9.63	64.65
Mixture-Based Decoder (using transfer learning)	-11.6	49.41	30.05
Autoregressive Transformer Decoder	-13.8	48.32	31.89

Chapter 5

CONCLUSION

In conclusion, our research has demonstrated significant advancements in object-centric representation learning for videos. By leveraging the memory-conditioned slot attention model, we achieved remarkable results in object discovery and multi-object tracking. Notably, our model outperformed the current state-of-the-art models, showcasing its effectiveness in capturing the temporal dynamics of video content.

The introduction of memory slots, capable of retaining information from previous frames, proved to be a pivotal aspect of our approach. This extension allowed our model to attend to past frames and incorporate temporal cues, ultimately enhancing object segmentation and multi-object tracking results. Moreover, we addressed the challenges posed by training Gated Recurrent Units (GRUs) over extended temporal sequences by using truncated backpropagation through time, which significantly improved our model's performance.

We conducted our experiments using the MOVI-D and MOVI-E datasets, which feature realistic scenarios with object motion and camera dynamics. The results obtained in terms of object segmentation and multi-object tracking were consistently encouraging. While our model represents a significant leap forward, there remains ample room for improvement. Despite its superiority in multi-object tracking over existing models, we acknowledge the need for further enhancements to achieve even higher accuracy and precision.

Our model works better when the objects have uniform patterns. The limitation of our model is that it splits complex objects into multiple slots. The size of the objects also matters; the model tends to discover and track medium-sized objects better than large-size objects. Large objects are frequently split into multiple slots, and small objects are usually confused with the components of another object. Apart from these limitations, the model performs generally well.

A possible future direction for this research could involve expanding it to encompass 3D data. Object-centric representation learning can be used in sequence-to-sequence generative models. Our approach will be useful for generating longer sequences by input of long sequences. Furthermore, our work is completely unsupervised, and weak supervision signals can be used for better performance.

BIBLIOGRAPHY

Anciukevicius, T., Lampert, C. H., and Henderson, P. (2020). Object-centric image generation with factored depths, locations, and appearances. *arXiv preprint arXiv:2004.00642*.

Ba, J. L., Kiros, J. R., and Hinton, G. E. (2016). Layer normalization. *arXiv preprint arXiv:1607.06450*.

Bahdanau, D., Cho, K., and Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Bernardin, K. and Stiefelhagen, R. (2008). Evaluating multiple object tracking performance: the clear mot metrics. *EURASIP Journal on Image and Video Processing*, 2008:1–10.

Besbinar, B. and Frossard, P. (2021). Self-supervision by prediction for object discovery in videos. In *2021 IEEE International Conference on Image Processing (ICIP)*, pages 1509–1513. IEEE.

Burgess, C. P., Matthey, L., Watters, N., Kabra, R., Higgins, I., Botvinick, M., and Lerchner, A. (2019). Monet: Unsupervised scene decomposition and representation. *arXiv preprint arXiv:1901.11390*.

Caron, M., Touvron, H., Misra, I., Jégou, H., Mairal, J., Bojanowski, P., and Joulin, A. (2021). Emerging properties in self-supervised vision transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 9650–9660.

Carvalho, W., Liang, A., Lee, K., Sohn, S., Lee, H., Lewis, R. L., and Singh, S. (2020). Reinforcement learning for sparse-reward object-interaction tasks in a first-person simulated 3d environment. *arXiv preprint arXiv:2010.15195*.

Chen, C., Deng, F., and Ahn, S. (2021). Roots: Object-centric representation and rendering of 3d scenes. *The Journal of Machine Learning Research*, 22(1):11770–11805.

Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014). Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

Crawford, E. and Pineau, J. (2019). Spatially invariant unsupervised object detection with convolutional neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 3412–3420.

Crawford, E. and Pineau, J. (2020a). Exploiting spatial invariance for scalable unsupervised object tracking. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 3684–3692.

Crawford, E. and Pineau, J. (2020b). Learning 3d object-oriented world models from unlabeled videos. In *Workshop on Object-Oriented Learning at ICML*.

Creswell, A., Kabra, R., Burgess, C., and Shanahan, M. (2021). Unsupervised object-based transition models for 3d partially observable environments. *Advances in Neural Information Processing Systems*, 34:27344–27355.

Creswell, A., Nikiforou, K., Vinyals, O., Saraiva, A., Kabra, R., Matthey, L., Burgess, C., Reynolds, M., Tanburn, R., Garnelo, M., et al. (2020). Alignnet: Unsupervised entity alignment. *arXiv preprint arXiv:2007.08973*.

Du, Y., Li, S., Sharma, Y., Tenenbaum, J., and Mordatch, I. (2021). Unsupervised learning of compositional energy concepts. *Advances in Neural Information Processing Systems*, 34:15608–15620.

Du, Y., Smith, K., Ulman, T., Tenenbaum, J., and Wu, J. (2020). Unsupervised discovery of 3d physical objects from video. *arXiv preprint arXiv:2007.12348*.

Elman, J. L. (1990). Finding structure in time. *Cognitive science*, 14(2):179–211.

Engelcke, M., Kosiorek, A. R., Jones, O. P., and Posner, I. (2019). Genesis: Generative scene inference and sampling with object-centric latent representations. *arXiv preprint arXiv:1907.13052*.

Eslami, S., Heess, N., Weber, T., Tassa, Y., Szepesvari, D., Hinton, G. E., et al. (2016). Attend, infer, repeat: Fast scene understanding with generative models. *Advances in neural information processing systems*, 29.

Greff, K., Belletti, F., Beyer, L., Doersch, C., Du, Y., Duckworth, D., Fleet, D. J., Gnanapragasam, D., Golemo, F., Herrmann, C., Kipf, T., Kundu, A., Lagun, D., Laradji, I., Liu, H.-T. D., Meyer, H., Miao, Y., Nowrouzezahrai, D., Oztireli, C., Pot, E., Radwan, N., Rebain, D., Sabour, S., Sajjadi, M. S. M., Sela, M., Sitzmann, V., Stone, A., Sun, D., Vora, S., Wang, Z., Wu, T., Yi, K. M., Zhong, F., and Tagliasacchi, A. (2022). Kubric: a scalable dataset generator.

Greff, K., Kaufman, R. L., Kabra, R., Watters, N., Burgess, C., Zoran, D., Matthey, L., Botvinick, M., and Lerchner, A. (2019). Multi-object representation learning with iterative variational inference. In *International Conference on Machine Learning*, pages 2424–2433. PMLR.

Greff, K., Rasmus, A., Berglund, M., Hao, T., Valpola, H., and Schmidhuber, J. (2016). Tagger: Deep unsupervised perceptual grouping. *Advances in Neural Information Processing Systems*, 29.

Greff, K., Van Steenkiste, S., and Schmidhuber, J. (2017). Neural expectation maximization. *Advances in Neural Information Processing Systems*, 30.

Greff, K., Van Steenkiste, S., and Schmidhuber, J. (2020). On the binding problem in artificial neural networks. *arXiv preprint arXiv:2012.05208*.

He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

He, Z., Li, J., Liu, D., He, H., and Barber, D. (2019). Tracking by animation: Unsupervised learning of multi-object attentive trackers. In *Proceedings of*

the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 1318–1327.

Henderson, P. and Lampert, C. H. (2020). Unsupervised object-centric video generation and decomposition in 3d. *Advances in Neural Information Processing Systems*, 33:3106–3117.

Im Im, D., Ahn, S., Memisevic, R., and Bengio, Y. (2017). Denoising criterion for variational auto-encoding framework. In *Proceedings of the AAAI conference on artificial intelligence*, volume 31.

Jang, E., Gu, S., and Poole, B. (2016). Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*.

Jiang, J., Janghorbani, S., De Melo, G., and Ahn, S. (2019). Scalor: Generative world models with scalable object representations. *arXiv preprint arXiv:1910.02384*.

Kabra, R., Zoran, D., Erdogan, G., Matthey, L., Creswell, A., Botvinick, M., Lerchner, A., and Burgess, C. (2021). Simone: View-invariant, temporally-abstracted object representations via unsupervised video decomposition. *Advances in Neural Information Processing Systems*, 34:20146–20159.

Kahneman, D., Treisman, A., and Gibbs, B. J. (1992). The reviewing of object files: Object-specific integration of information. *Cognitive psychology*, 24(2):175–219.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kipf, T., Elsayed, G. F., Mahendran, A., Stone, A., Sabour, S., Heigold, G., Jonschkowski, R., Dosovitskiy, A., and Greff, K. (2021). Conditional object-centric learning from video. *arXiv preprint arXiv:2111.12594*.

Kipf, T., Van der Pol, E., and Welling, M. (2019). Contrastive learning of structured world models. *arXiv preprint arXiv:1911.12247*.

Kosiorek, A., Kim, H., Teh, Y. W., and Posner, I. (2018). Sequential attend, infer, repeat: Generative modelling of moving objects. *Advances in Neural Information Processing Systems*, 31.

Lamb, A., He, D., Goyal, A., Ke, G., Liao, C.-F., Ravanelli, M., and Bengio, Y. (2021). Transformers with competitive ensembles of independent mechanisms. *arXiv preprint arXiv:2103.00336*.

Lin, Z., Wu, Y.-F., Peri, S., Fu, B., Jiang, J., and Ahn, S. (2020a). Improving generative imagination in object-centric world models. In *International Conference on Machine Learning*, pages 6140–6149. PMLR.

Lin, Z., Wu, Y.-F., Peri, S. V., Sun, W., Singh, G., Deng, F., Jiang, J., and Ahn, S. (2020b). Space: Unsupervised object-oriented scene representation via spatial attention and decomposition. *arXiv preprint arXiv:2001.02407*.

Locatello, F., Weissenborn, D., Unterthiner, T., Mahendran, A., Heigold, G., Uszkoreit, J., Dosovitskiy, A., and Kipf, T. (2020). Object-centric learning with slot attention.

Löwe, S., Greff, K., Jonschkowski, R., Dosovitskiy, A., and Kipf, T. (2020). Learning object-centric video models by contrasting sets. *arXiv preprint arXiv:2011.10287*.

Löwe, S., Lippe, P., Rudolph, M., and Welling, M. (2022). Complex-valued autoencoders for object discovery. *arXiv preprint arXiv:2204.02075*.

Luong, M.-T., Pham, H., and Manning, C. D. (2015). Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*.

Milan, A., Leal-Taixé, L., Reid, I., Roth, S., and Schindler, K. (2016). Mot16: A benchmark for multi-object tracking. *arXiv preprint arXiv:1603.00831*.

Rand, W. M. (1971). Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical association*, 66(336):846–850.

Savarese, P., Kim, S. S., Maire, M., Shakhnarovich, G., and McAllester, D. (2021). Information-theoretic segmentation by inpainting error maximization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4029–4039.

Singh, G., Deng, F., and Ahn, S. (2022a). Illiterate DALL-e learns to compose. In *International Conference on Learning Representations*.

Singh, G., Peri, S., Kim, J., Kim, H., and Ahn, S. (2021). Structured world belief for reinforcement learning in pomdp. In *International Conference on Machine Learning*, pages 9744–9755. PMLR.

Singh, G., Wu, Y.-F., and Ahn, S. (2022b). Simple unsupervised object-centric learning for complex and naturalistic videos. *Advances in Neural Information Processing Systems*, 35:18181–18196.

Spelke, E. S. and Kinzler, K. D. (2007). Core knowledge. *Developmental science*, 10(1):89–96.

Stanić, A. and Schmidhuber, J. (2019). R-sqair: Relational sequential attend, infer, repeat. *arXiv preprint arXiv:1910.05231*.

Stelzner, K., Kersting, K., and Kosiorek, A. R. (2021). Decomposing 3d scenes into objects via unsupervised volume segmentation. *arXiv preprint arXiv:2104.01148*.

Van Steenkiste, S., Chang, M., Greff, K., and Schmidhuber, J. (2018). Relational neural expectation maximization: Unsupervised discovery of objects and their interactions. *arXiv preprint arXiv:1802.10353*.

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

Veerapaneni, R., Co-Reyes, J. D., Chang, M., Janner, M., Finn, C., Wu, J., Tenenbaum, J., and Levine, S. (2020). Entity abstraction in visual model-based reinforcement learning. In *Conference on Robot Learning*, pages 1439–1456. PMLR.

Wang, Y., Shen, X., Hu, S. X., Yuan, Y., Crowley, J. L., and Vaufreydaz, D. (2022). Self-supervised transformers for unsupervised object discovery using normalized cut. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14543–14553.

Watters, N., Matthey, L., Bosnjak, M., Burgess, C. P., and Lerchner, A. (2019a). Cobra: Data-efficient model-based rl through unsupervised object discovery and curiosity-driven exploration. *arXiv preprint arXiv:1905.09275*.

Watters, N., Matthey, L., Burgess, C. P., and Lerchner, A. (2019b). Spatial broadcast decoder: A simple architecture for learning disentangled representations in vaes. *arXiv preprint arXiv:1901.07017*.

Weis, M. A., Chitta, K., Sharma, Y., Brendel, W., Bethge, M., Geiger, A., and Ecker, A. S. (2021). Benchmarking unsupervised object representations for video sequences. *The Journal of Machine Learning Research*, 22(1):8253–8313.

Williams, R. J. and Peng, J. (1990). An efficient gradient-based algorithm for on-line training of recurrent network trajectories. *Neural computation*, 2(4):490–501.

Wu, Y.-F., Yoon, J., and Ahn, S. (2021). Generative video transformer: Can objects be the words? In *International Conference on Machine Learning*, pages 11307–11318. PMLR.

Yang, Y., Chen, Y., and Soatto, S. (2020). Learning to manipulate individual objects in an image. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6558–6567.

Yu, P., Xie, S., Ma, X., Zhu, Y., Wu, Y. N., and Zhu, S.-C. (2021). Unsupervised foreground extraction via deep region competition. *Advances in Neural Information Processing Systems*, 34:14264–14279.

Zoran, D., Kabra, R., Lerchner, A., and Rezende, D. J. (2021). Parts: Unsupervised segmentation with slots, attention and independence maximization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10439–10447.

Appendix A

QUALITATIVE RESULTS FOR FULL SEQUENCE

In this section, we provide additional visual results of our model and the baseline models for comparison. In the figures, all 24 frames can be shown along with their reconstruction and object masks. Notice the object segmentation and object tracking throughout the sequence.

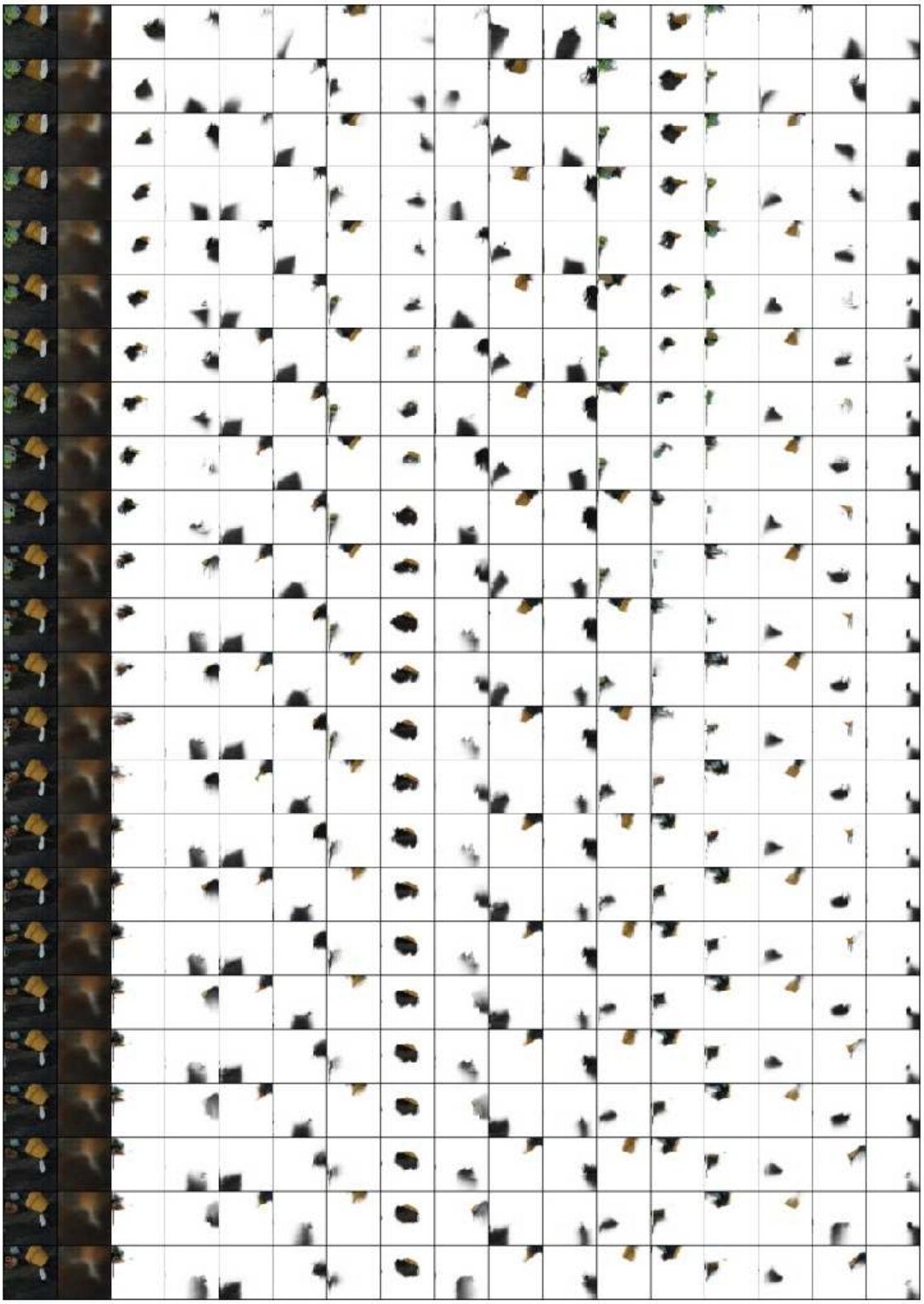


Figure A.1: SAVi Results on a Sample from MOVI-D. The first column is the input frames, the second column is the full reconstruction and the following columns are the slot reconstructions.

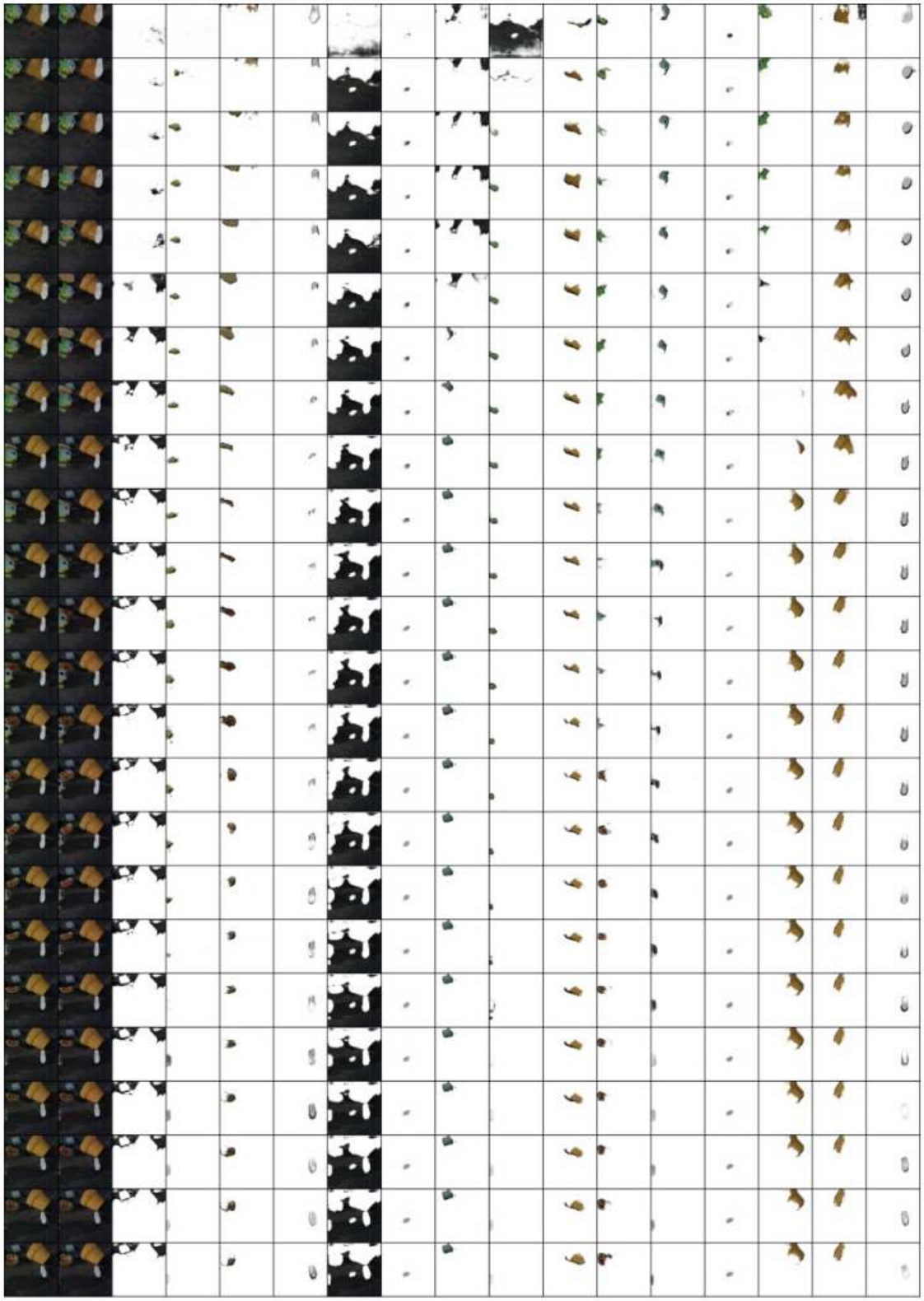


Figure A.2: STEVE Results on a Sample from MOVI-D. The first column is the input frames, the second column is the full reconstruction, and the following columns are the slot reconstructions.

Figure A.3: Our Model Results on a Sample from MOVI-D. The first column is the input frames, the second column is the full reconstruction, and the following columns are the slot reconstructions.