Rheinische
Friedrich-Wilhelms-
Universität Bonn

Institute for Computer Science
Department VI
Autonomous Intelligent Systems

RHEINISCHE
FRIEDRICH-WILHELMS-UNIVERSITÄT BONN

MASTER THESIS

Learning Object Dynamics and Interactions for Object-Centric Video Prediction

Author:
Ismail WAHDAN

First Examiner:
Prof. Dr. Sven BEHNKE

Second Examiner:
PD Dr. Volker STEINHAGE

Date: February 9, 2023

Declaration

I hereby declare that I am the sole author of this thesis and that none other than the specified sources and aids have been used. Passages and figures quoted from other works have been marked with appropriate mention of the source.

Place, Date

Signature

Abstract

Extracting the compositional structure of a scene, as well as modeling the dynamics of different objects, are desired properties for autonomous agents' planning and reasoning capabilities. While existing methods can successfully decompose a scene into its object components in an unsupervised manner, understanding dynamics and object interactions from visual observations still remains a challenging task. In this work, we extend the popular Slot Attention model and its extension Slot Attention for Video (SAVi) model for the task of object-centric video prediction, i.e., modeling the spatio-temporal dynamics of objects in a video in order to predict the future object states, from which we can then generate subsequent video frames. With the goal of learning meaningful object-centric spatio-temporal representations, we perform an in-depth analysis of the role of different predictor modules, including recurrent models and transformers, for modeling object dynamics and generating realistic future frames. We also introduce two novel transformer-based architectures specific for this task. In our experiments, we show promising results indicating that our predictor learns to represent different objects, model object dynamics, and understand their significance for the task of future prediction. Furthermore, we show that the learned object-centric representations can be used for future frame prediction providing good, interpretable results.

Contents

1. Introduction

Understanding a scene is a much more complicated process than it seems at first glance. It is not just about recognizing what is where, but about recognizing actions taking place, possible next states, and the relations and interactions between different objects in the scene. Each of these tasks are well known problems in the fields of machine learning and computer vision. Segmentation tries to answer the what and where questions, video prediction addresses the task of understanding what happens next in the scene, and object-centric learning tries to understand the relations and interactions between existing entities in the scene.

Computers see images and videos as discrete pixel values, whereas human brains are optimized to perceive the world in terms of objects (when discrete and countable) or stuff (Green and Quilty-Dunn 2021; S. P. Johnson 2018; Kahneman, Treisman, and Gibbs 1992; Spelke and Kinzler 2007). Bridging the gap between the two representations would make it easier and more efficient for computers to understand the actions and interactions in the world, in the same way it is easier to compute the dynamics of a moving ball as a whole than computing the dynamics of each thread holding it together. Moreover, understanding the world in the same way as humans do will help computers give clearer and explainable answers to humans. Machine learning algorithms, which are used as black box methods, face obstacles on the way for mainstream use in fields where actions need to be well-interpreted and justified (Adadi and Berrada 2018; Char, Shah, and Magnus 2018; Goodman and Flaxman 2017; Samek, Wiegand, and Müller 2017).

In this thesis, we build upon an existing model for object-centric learning, the Slot Attention (SA) model (Locatello et al. 2020), and later move on the following up model of Slot Attention for Videos (SAVi) (Kipf, Elsayed, et al. 2022), for reasons that will be discussed throughout the thesis. We extend both of them for the pretext task of object-centric video prediction. Our goal is to improve the object representations learned by the models and have predictor models that are able to represent relations between said objects. Such relations need temporal information to be identified, which is why we deal with sequence processing and video prediction.

Another character of intelligent agents is the ability to expect results of certain actions and interactions. Video prediction is the equivalent problem in the field of computer vision. The task is defined as such: given n context frames, what is a possible future in the next m frames. Due to the stochasticity of the future, sometimes just a reasonable or the most likely output is desired, and sometimes the approach tries to model several possibilities of the future. However, video prediction is rarely sought after for its own solution, but because it provides the valuable option to learn data representations in a self-supervised manner, without the need for labeling and extensive human labour. One video is trivially divided into a seed/context video segment that is used as input, and the following frames are the desired output. Most of the effort here goes into designing an architecture that learns the aspects that we want the model to be able to identify and extract. In our case, these aspects are distinguishing objects, encoding them with enough information to be able to reconstruct them, and to be able to model dynamics and interactions between such objects and predict their upcoming states. These aspects are what defined object-centric video prediction. The model tries to extract existing objects in a scene, understand their current states and interactions, then produce an output of prediction for each object. This output is reconstructed back to a common scene which presents the predicted frame. In figure 1.1, we see an introductory figure of this thesis's approach that demonstrates this process.

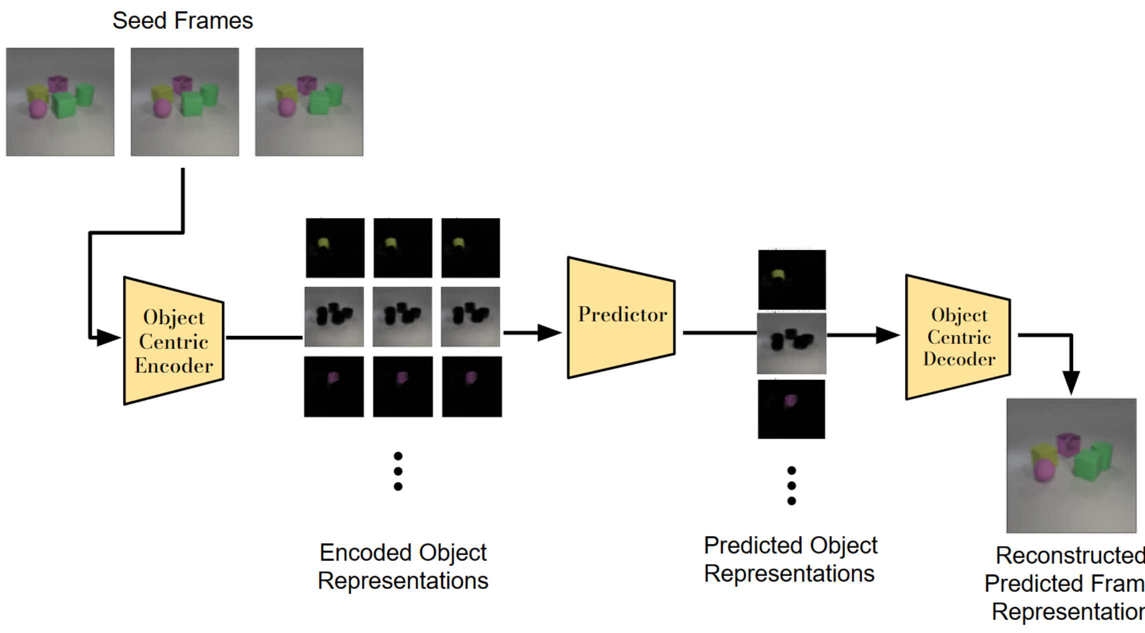
Which prediction paradigms to use in an object-centric approach to video prediction is not an easy question to answer. A good predictor will need to capture a lot of information from mixed facets. It needs to capture information about the object and its history, as well as its interactions with the environment and other objects. All of this has to be done efficiently. This thesis tries to answer this question by comparing these paradigms in the same setting, which is possible by separating the object-centric representation learning for the actual prediction. As such, this is our approach.

Our main contribution in this thesis can be summerized with the following points:

- Proposal of a deep learning pipeline that utilizes object-centric representations, learned by slot-attention based models, to try and predict future states of objects in the scene, as well as the future of the scene as a whole.

- Investigation of different predictors and introduction of two novel object-centric transformers that predict future object states in a more structured and interpretable manner than a regular transformer model.

- Evaluation of the different predictors and perform a number of ablation studies in pursuit of the best training setup, and we show through our evaluation that the predictors we are using are able to deliver meaningful results.


Seed Frames
Object Centric Encoder
Predictor
Object Centric Decoder
Encoded Object Representations
Predicted Object Representations
Reconstructed Predicted Frame Representation
Figure 1.1: Example of object-centric deconstruction and reconstruction of a scene for the purpose of frame prediction, as performed by the approach proposed in this thesis. The scene is encoded into object representations. Prediction is then performed on these representations. The result is combined and decoded back to generate the predicted frame.

The thesis is organized as follows: Chapter 2 will discuss the fundamentals, including general computer vision and deep learning topics needed for the understanding of this thesis. Chapter 3 will discuss the related work and current status of research in video prediction and object-centric learning. Chapter 4 will introduce our methodology and all contributions of the thesis. Chapter 5 will include discussions of the experimental setup, model evaluation and various experiments and their results. Finally, chapter 6 will include the conclusion and any discussions about the future work.

2. Fundamentals

In this chapter we present general topics of computer vision, deep learning, and sequence processing that are of interest for this thesis. We start by introducing multiple neural networks paradigms that are relevant for processing of visual and sequential data, starting by convolution, the main method of extracting visual information in neural network, then by discussing encoders and decoders and their meaning for sequence processing, then move on to Recurrent Neural Networks (RNNs), taking the example of Long Short Term Memory (LSTM) (Hochreiter and Schmidhuber 1997) architecture. We then present the more recent alternative transformer networks and discuss their building units, including the self-attention mechanism and positional encoding.

2.1. Convolution and Convolutional Neural Networks

In general, convolution is the operation of applying change on a signal using another one. For computer vision, convolutions are two dimensional and discrete, and their kernels are usually described as filters. In classical computer vision, filters were handcrafted for certain tasks, such as edge detection and noise cancellation. In deep learning, filter weights are model parameters learned in a data-driven manner by back-propagation. Mathematically, convolution for images is described by:

$$I(x, y) * F(x, y) = \sum_{v \in -1,0,1} \sum_{u \in -1,0,1} I(x, y).F(x - v, y - u) \tag{2.1}$$

In modern day networks, convolutions as building units are used in bulk and combined in different ways. They can be cascaded, stacked, have skip connections and combined in parallel (He et al. 2016; Huang et al. 2017; Simonyan and Zisserman 2015; Szegedy et al. 2015). Non-linear activation functions are used between layers to prevent the model from collapsing and to allow for learning non-linear functions. To avoid depth problems, skip connections are used, as well as dropout and different normalization techniques. In figure 2.1, we see an example of a standard neural network with stack layers and non-linear activation functions.

In most of the mentioned architectures above, the spatial resolution of the fea-

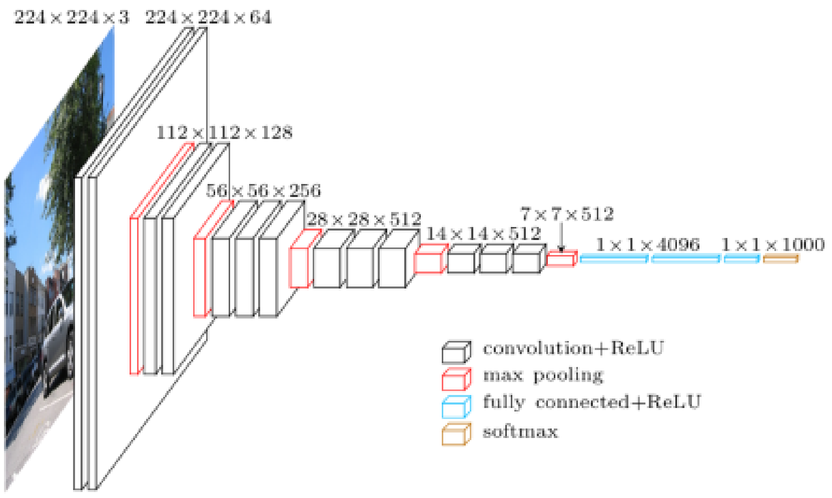


Figure 2.1: Diagram of a standard VGG16 convolutional architecture. (Simonyan and Zisserman 2015).

ture maps (outputs of one layer passed as input to the next) decreases as the network grows deeper. This downsampling is achieved via mechanisms such as average or max pooling, or strided convolutions. These mechanisms aim to reduce the number of passes over the input, and to encourage the network to learn how to abstract the information. However, for applications where the desired output is in the original input's size, e.g. semantic or instance segmentation, the output needs to be resized back by a decoding phase, that upsamples the feature maps till it reaches the desired size (Badrinarayanan, Kendall, and Cipolla 2017; Ronneberger, Fischer, and Brox 2015; H. Wu et al. 2019). Transposed convolutions and interpolation filters are among the popular upsampling techniques used there. In figure 2.2, we see an example of the popular U-Net architecture for image segmentation, that utilizes upsampling as will as skip connections to rebuild the image back in a hierarchical manner,

Instead of always employing randomly initialized models, convolution models already trained on large datasets (Deng et al. 2009; T.-Y. Lin et al. 2014), e.g. pre-trained ResNet (He et al. 2016) models, are often used for feature extraction, as they have learned to extract most relevant information from visual content. Most often outputs of the first few layers are the most useful and generalizable, and they are often called feature maps. This operation of using the parameters learned on a task to extract information for another one is known as transfer learning, and it can be used in the context of object-centricity to extract and describe objects in an image.

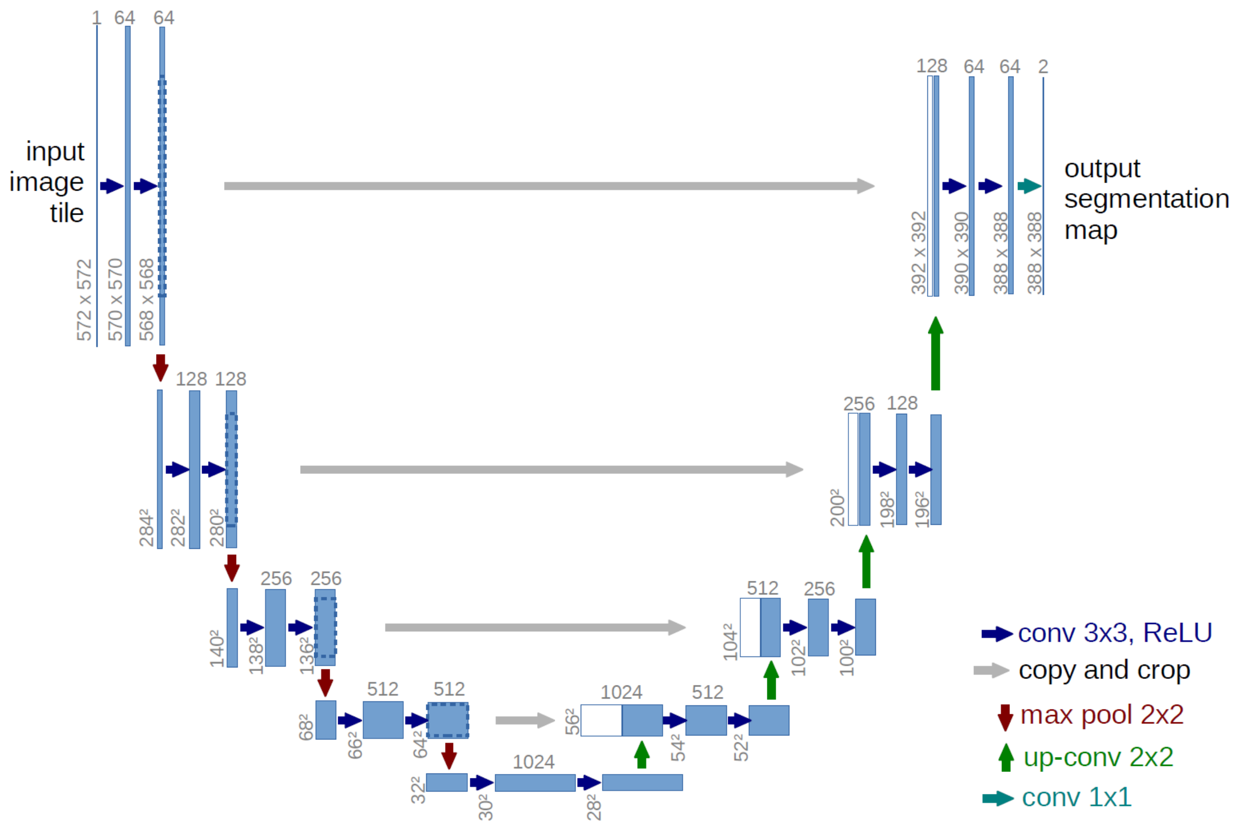


Figure 2.2: Diagram of a U-Net architecture for image segmenation. (Ronneberger, Fischer, and Brox 2015).

2.2. Encoder-Decoder Architecture

An Encoder-Decoder architecture is a type of neural network that, as the name suggests, consists of two components: the encoder, and the decoder. The encoder's job is to transfer (encode) the input into a hidden space, that represents its information in another format. Usually it is desired that this space is smaller than the original space, so that the encoder learns to extract the most relevant information to its task, e.g. semantics of objects. Decoders then translate back this hidden representation (decodes it) into a desired output space. This output space can be the original space, as in the case with Autoencoders, where the target is learning this process itself and being able to store the information in the hidden space. A currently more common use case is decoding the encoded representations into another space that is easier to reach from the hidden space. One example here is semantic segmentation, as shown in the convolution section, where the encoder (downsampling) part learns an abstraction of the image parts and what they include, while the decoder turns this information into segmentation of the image.

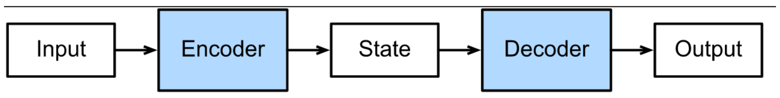


Figure 2.3: Encoder-decoder structure. (A. Zhang et al. 2021).

Another example is image denoising, the task of regenerating the image without signal noise in it. Such task also use similar architectures to image segmentation, and by going first to the hidden space that has more information density, noise can be removed since the model can learn it is not a relevant or desired of the information it needs to encode. The decoder's task is then to reconstruct this purer represenation of the image.

The encoder-decoder paradigm is also often used in sequence processing, because mapping a sequence to another sequence is a hard task to learn. It can be too complicated to directly match elements of two sequences together and understand their relations, so instead the hidden space is used as a middle ground, such that the encoder learns to capture the meaning of a sequence in this intermediate space, whereas the decoder learns how to represent this meaning in the desired space, improving the learning capabilities of sequence-to-sequence models.

An encoder could also take the input apart and be built to extract certain information from it. For example, in this thesis, we deal with slot attention, a mechanism that encodes an image in a set of the vectors, each assigned to an object in the image. In this case, the goal of the encoder is to extract and seperate objects in the image, with the goal of learning an object-centric view of the world.

2.3. Recurrent Neural Networks

Recurrent Neural Networks (RNNs) are a family of deep learning models built for various tasks of sequence processing, including many-to-one problems, e.g. sequence labelling (Akbik, Blythe, and Vollgraf 2018; Ma and Hovy 2016; Namysl, Behnke, and Köhler 2020), one-to-many problems , e.g. image captioning (Herdade et al. 2019; Hossain et al. 2019; Vinyals et al. 2016), and many-to-many problems, e.g. machine translation (M. X. Chen et al. 2018; K. Cho, Van Merriënboer, Bahdanau, et al. 2014; K. Cho, Van Merriënboer, Gulcehre, et al. 2014) or video prediction (Farazi, Nogga, and Behnke 2021; Finn, Goodfellow, and Levine 2016; Karapetyan et al. 2022; Oprea et al. 2020; Poibrenski et al. 2020; Santana and Hotz 2016; Y.-F. Wu, Yoon, and Ahn 2021; Ye et al. 2019; Zang et al. 2022).

Dealing with sequences requires the model to have notion of history and the previous inputs it received. There are different types of RNNs that try to manage

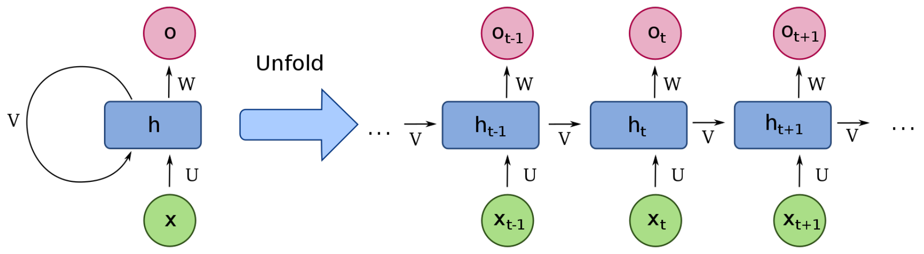


Figure 2.4: Simple RNN structure. (Deloche 2017).

this differently, however, most of them, including the most basic and the most popular forms, do this by introducing a vector h, called the hidden state. In the most basic form of RNNs, shown in figure 2.4, the network consist of neurons with an input, output, and self-connections. The core in RNNs is that the output h of the self-connection, should preserve information of previous inputs. For example, in a task of machine translation, h would preserve information contained in previous words in the sentence, like genders, counts, and tenses, which are all needed while choosing the next output.

Learning to store this information is done by two things: the model uses this hidden state in computing the output, so the output does not just depend on the current input, but also on the hidden state summarizing the previous sequence elements it saw. Secondly, the model performs back-propogation through time, so previous output and hidden states are all involved in the learning process and parameter optimization, so the history is included in the learning. Thus, RNNs depend on parameter sharing in a sense, because the output and the hidden state for all time steps are all generated using the same parameters, so these parameters have to optimize for all sequence steps and not just any particular one of it.

The problem that arises here is that storing all this information in one vector is hard, and often leads to information loss. Moreover, while learning, the elements further back in time gradually lose the ability to affect the gradient, because of the problem known as vanishing gradient (Hochreiter and Schmidhuber 1997; Pascanu, Mikolov, and Bengio 2013), which describes how gradients in the earlier layers of any model have very small effect on learning. The analogy of unfolding the network, as shown in the figure, shows that going back in time is similar to back-propagating back to earlier layers in a deep model. Such problems lead to the introduction of types of RNNs that try to migitate them, including LSTMs.

Long Short Term Memory (LSTM) models is a type of RNNs that addresses the above mentioned problems by trying to control what the model remembers and what it forgets. Unlike simple RNNs, LSTMs keep track of two vectors over time, h and C. h is the cell output that is reused in the following next step, representing the short memory part, and C is the cell state that keeps tracks of cell history as a whole, with less changes between time steps, representing the long memory part.

LSTMs are built out of gates, which are ways to continue information flow to the cell state. A gate consists of a linear mapping, followed by an activation function. It has three gates, as shown in the structure in figure 2.5, each named according to their functionalities. The first gate is the forget gate, that looks at current input x_t and last output h_t, and decides what information is likely not important any more, and can be discarded from the cell state, freeing up space for more important information. This done through a sigmoid function that outputs a value between 0 and 1 for each element in the C vector. Multiplication with these values decide what is kept and what is forgotten, with multiplication with 1 acting as completely keep, and with 0 as completely forget. The output of this gate is f_t.

The second gate is the input gate, which is another sigmoid that decides what part of the new input will be needed for the future, and as such will be remembered by the cell state. The sigmoid result i_t acts in a similar way as with the forget gate, specifying the magnitude of "remembrance", while the *tanh* output c_t specifies the direction, i.e., will information be deleted or added, since the *tanh* output is in the range $[-1, 1]$.

The last gate is the output gate, which specifies in a similar manner how the cell state could be changed to produce the new output. h_t is used either directly as an output, or it can be processed by another feed forward model. Both states are passed to the next time step, and the operation is repeated.

LSTMs variations are still among the most used models for sequence processing in different fields, and in our thesis we use them as a baseline in prediction to compare new approaches with.

2.4. Transformers

Transformer networks (Vaswani et al. 2017) were presented in 2017 as an alternative architecture for sequence processing. It is a multilayered architecture that uses self-attention to be able to process long sequential data, without having to deal with the memory loss issues faced by RNNs. Transformers also have the advantage of being able to process elements of sequences in parallel, which make them far more efficient for longer sequences. Transformers were first introduced for and applied in the field of natural language processing (NLP) and were later adapted to computer vision applications.

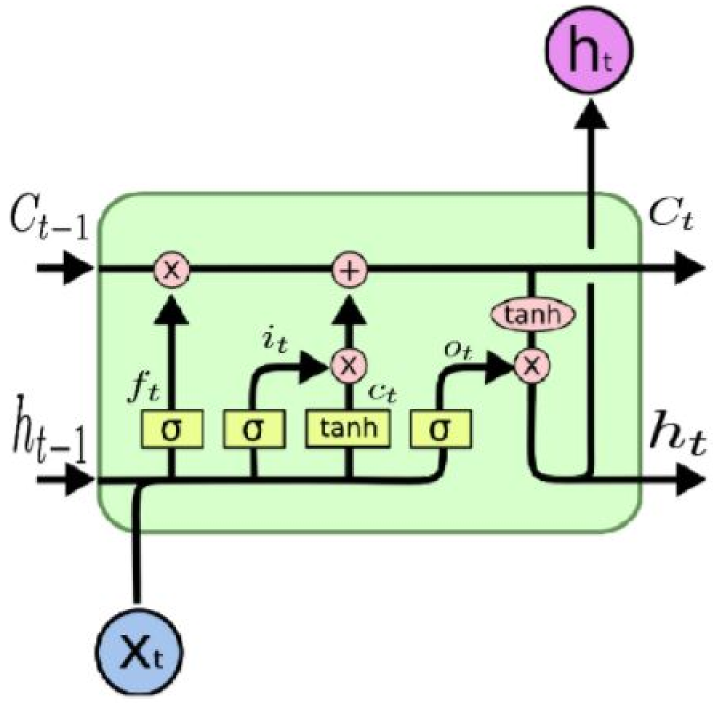


Figure 2.5: Overview of an LSTM cell (Olah 2015).

2.4.1. Architecture Overview of Transformers

A general overview of the transformer structure can be seen in figure 2.6. Transformers are an encoder-decoder architecture, where depending on the application, encoders or decoders are used as stand-alone models, or together.

In the beginning, the feature maps are embedded to a suitable vector space to be used in transformers. Transformers need to be applied to sequences of one-dimensional vectors and not directly on images. Since the attention operation allows every vector to attend to other vectors, a positional embedding is added to the vectors to preserve temporal information. There are many ways to encode position. The encoding that was used in the original work, as well as in this thesis, is:

$$\begin{aligned} PE_{(pos,2i)} &= \sin(pos/10000^{2i/d_{model}}) \\ PE_{(pos,2i+1)} &= \cos(pos/10000^{2i/d_{model}}) \end{aligned} \tag{2.2}$$

where PE is the vector that will be added to the input according to position, and d_{model} is the embedding dimension.

Attention, as shown in the figure, is a simple scaled dot product process of three components, query Q, value V and key K. This key/value/query concept

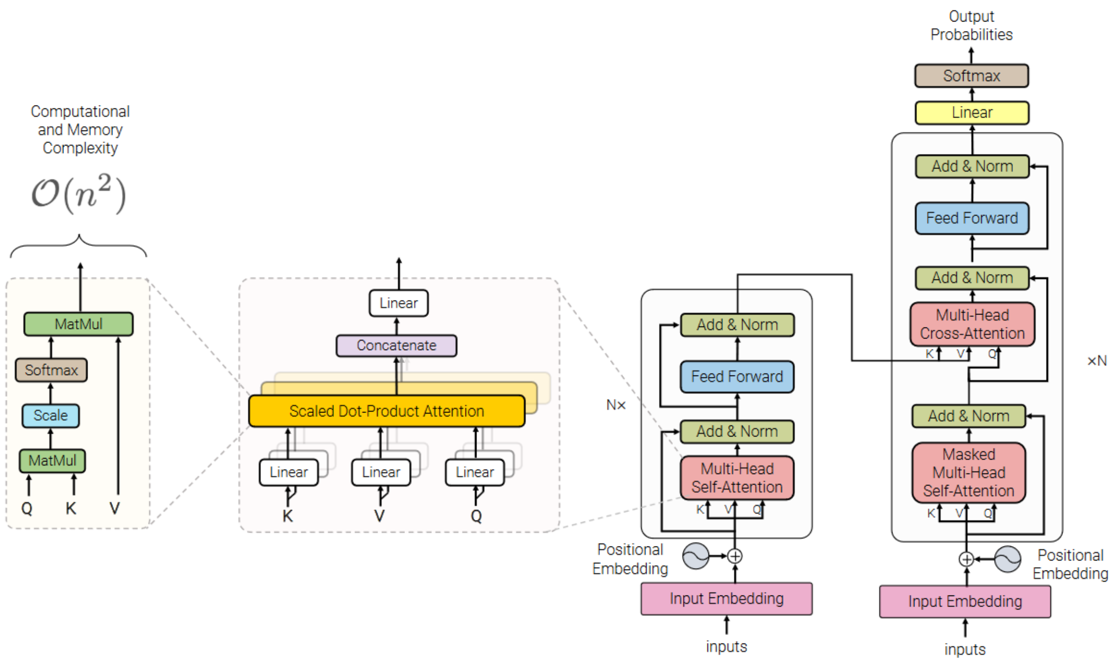


Figure 2.6: Transformer architecture. (Tay et al. 2020).

comes from the field of information retrieval, where a query is compared with a set of keys to produce the best matching values. In the case of transformers, all three components are a result of a learned linear projection of the input features themselves. The goal is to learn which features should attend to which features depending on their content. Mathematically, attention is expressed as:

$$ATTENTION(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V \tag{2.3}$$

where d_k is dimension of keys, and is used to prevent the parameter inside the softmax function from growing to a scale where the function has negligible values.

The rest of the architecture consists of regular building units of neural architectures, such as residual connections, layer normalization, and feed forward connections. The decoder has similar parts, with the notable introduction of masked self-attention, that assures that the tokens can only look back in time and can't base their output on the future. We also see the cross-attention layer, also known as encoder-decoder-attention, where the three components actually differ from each other. Keys and value come from encoder, and the query from the decoder, resulting in a mapping between the encoder and decoder outputs.

2.4.2. Transformers for Computer Vision

Although images are not usually seen as sequences, the high performance of transformers and their ability to surpass RNNs begged the questions, whether they would be able to do the same to convolution and CNNs. Vision Transformers (ViT) (Dosovitskiy et al. 2021) were introduced to do this. They turn an image into a sequence of its patches, then embed their flattened form and use it as an input for a transformer encoder. The output of the encoder is passed then to an MLP classifier for the task of image classification.

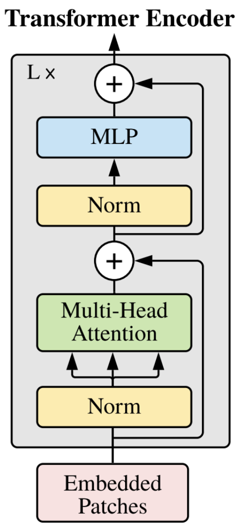


Figure 2.7: Transformer encoder used in vision transformers. (Dosovitskiy et al. 2021).

For this thesis, the important part is the encoder structure as shown in 2.7, since it is similar to what we will be using later in our model.

3. Related Work

3.1. Video Prediction

Making an intelligent decision depends on the ability on knowing the possible results of different decisions. An example is shown in figure 3.1. Given n context frames, m frames are predicted and used in decision making. Similar scenarios in autonomous systems and robotics have motivated the research in the video prediction problem as a goal in itself in the past years (Finn, Goodfellow, and Levine 2016; Karapetyan et al. 2022; Poibrenski et al. 2020; Santana and Hotz 2016; Zang et al. 2022). Furthermore, prediction tasks on videos, whether direct prediction of frames or other features, such as context, object tracking and rotation, have proven to be good pretext tasks for learning visual representations (Behrmann, Gall, and Noroozi 2021; Benaim et al. 2020; Diba et al. 2019; Farazi, Nogga, et al. 2021; Jing et al. 2018; Kim, D. Cho, and Kweon 2019; J. Wang, Jiao, and Liu 2020).

There have been various approaches for video prediction, which can be broadly classified under the three classes: convolutional, recurrent and generative models (Oprea et al. 2020). However, it is usually the case that many approaches combine these classes to solve each one's problems with the other.

Convolution on its own has serious limitations that prevent it from performing well enough in videos and in video prediction. The first is the limited perception of a kernel due to its size, as most kernels perceive smaller parts of an image or a frame, usually 3x3 or 5x5 blocks. This is usually mitigated by increasing the depth of the model and the downsampling of the feature maps. The second part is the inability to capture dependencies between frames and what is happening between them. For this, 3D convolutional networks are introduced, where time is another dimension in the kernel. However, while there are workarounds, and some works try to use purely convolutional models, such as (Gao et al. 2022) and (Chiu, Adeli, and Niebles 2020), most works that use them combine them with the use of other paradigms, usually RNNs. RNN are able to help the convolutional components by modeling temporal changes of visual features, which gives the model the ability to understand the evolution of the scene.

Using RNNs and other sequence processing paradigms is the most intuitive

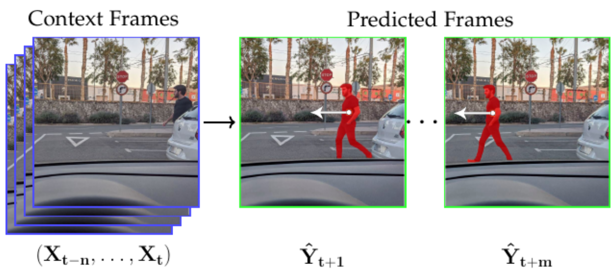


Figure 3.1: An example of a scenario where video prediction is needed from (Oprea et al. 2020). A person suddenly appears from behind the white car in front of the agent (a moving car). The agent has to expect that this person is going to walk in front of it and has to adjust its plan accordingly.

approach for video prediction, since a video is a sequence of visual features. The prediction itself is recursive in its nature, which is another plus point for recurrent models, as they are explicitly trained to be able to handle their own output as an input. Thus, there is an abundance of models that are built this way (Denton and Fergus 2018; Finn, Goodfellow, and Levine 2016; Karapetyan et al. 2022; Poibrenski et al. 2020; Santana and Hotz 2016; Y. Wang et al. 2017; Y.-F. Wu, Yoon, and Ahn 2021; Yan et al. 2022; Ye et al. 2019; Zang et al. 2022).

Recurrant models of course do not just use RNNs, but it is a core part of their structure. Convolutions are still used for feature extraction and understanding visual input, and generative networks are still used to try to handle the multi-modality of the future. An example here is the Stochastic Video Generation (SVG) model(Denton and Fergus 2018), which is a pixel-level prediction model that uses both a deterministic frame predictor and time-dependent stochastic latent variables. The predictor is a regular ConvLSTM (Shi et al. 2015) based encoder-decoder model. It is trained with the help of a pre-trained inference model that generates the latent variables, either from fixed (SVG-FP) or a learned (SVG-LP) prior. These latent variables should carry the stochastic information about the next frame that a deterministic model cannot capture. At inference time the prior is used directly without the need for the inference model. This approach shows that recurrent approaches on their own cannot model a multi-modal future, because the recurrent part is deterministic and a stochastic component needs to be introduced.

Another example of recurrent models is the Patch-based Object-centric Video Transformer (POVT) (Yan et al. 2022). It does not use regular RNNs, but transformer networks for modeling object dynamics. Transformers can fall under the umbrella of recurrent approaches, even though they are not RNNs, as they look at the whole input at once, but they too deal with sequence processing and see the input as a sequence. Most approaches that work with RNNs can work with transformers too, given the right adjustments. Transformers might even be the better alternative, depending on the application, because they can be better are modelling relations and interactions between objects, since again they have access to the whole input.

Another approach that can also be frequently, is the one seen in the conditional video prediction model used by (Kossen et al. 2019; Z. Lin, Y.-F. Wu, S. Peri, et al. 2020a; Ye et al. 2019). Here it uses graphs and graph neural networks (Kipf, Fetaya, et al. 2018) for modeling object representations. Edges in the graph represent the relations and interactions between objects, and can carry feature representations of them. However, these model often need supervision in locating and describing the entities and their relation to each other at least in the first frame, so the approach is not fully self-supervised.

For deeper understanding of the video prediction taks, it is worth it to take a look on the approaches of Frequency Domain Transformer Networks (FDTN) (Farazi and Behnke 2020) and Local Frequency Domain Transformer Networks (LFDTN) (Farazi, Nogga, and Behnke 2021). Here it is talked about the uniquely aims for separation of the three components of video prediction: interpretation of the formed internal representations, extracting transformations from input frames, projecting them onto the future frames by applying the transform on the current input. These mentioned approaches uniquely aim to compartmentalize these three components in the model. This allows for higher interpretability of the model, while having much fewer learnable parameters. In FDTN, a motion segmentation model that does not have any learnable parameters is introduced, which is inspired by classical linear dynamical systems theory and Kalman filters. This takes places in iterations of prediction and correction steps, and models foreground and background separately. The model is extended by a simple CNN to allow for learning foreground motion. In LFDTN, this is extended further to video prediction and not just motion prediction. These approaches allow us to analyse the different aspects of video prediction and understand what kind of work could be applied to each component.

For the object-centric case, these transformations are learned by objects. For this to be possible, we need to track the object over time and be able to match it with its own history. In the paper by (Y.-F. Wu, Yoon, and Ahn 2021), this issue is addressed. It introduces another example of a model that mixes generative and recurrent paradigms for video prediction, called Object-Centric Video Transformer (OCVT). The paper argues for the use of object tokenization in videos, similar to practices in the field of natural language processing, while keeping in mind that unlike words in a sentence, objects in a video do not have a natural order. Instead of serializing and predicting an ordered output, the whole image is predicted at once.

The proposed architecture consists of an encoder, in the form of a Variational Autoencoder (VAE) (Kingma and Welling 2013), that generates latent variables designed to be object-centric with explicit bounding box information. These latents are then passed as input to a transformer decoder, which generates the predicted latents at the next time step. To generate the reconstruction of the image, the VAE's decoder is used. The latent variables have four components: a binary value indicating the presence of an object, the boundary box of the object, the depth of the object, and all other information about the object. Each frame is divided into $H \times W$ cells, each of which detects the above latent variables, enabling the detection of $H \times W$ objects. The decoder uses deconvolution to create an image for each object.

As the model directly applies loss functions in the token space, it is necessary to correctly align objects in adjacent frames, to apply the loss function on tokens that represent the same object. As the output is not ordered, one object could be described by the first token at one time step, and the last token in the next. As such, the objects in both latent outputs are aligned using a permutation matrix and the Hungarian algorithm, to find an optimal matching. Other alignment strategies, such as incorporating appearance information and learning the permutation matrix, may also be used. This alignment idea is highly relevant to the work of this thesis, especially at earlier stages as we used Slot Attention before SAVi (both will be discussed in the next section).

Unlike most of the work mentioned here, the work of this thesis is highly interpretable by the using of attention mechanisms that allows investigation of how object states and interactions affect prediction. Object interactions are modeled intuitively through attention over interacting objects and over past states, and the attention is modeled efficiently by separating temporal and object attentions. There is no need to use specially-designed structures like graphs. There is no need to use a number of components equal to the number of possible interactions, because interactions are modelled as a relation between objects, and not a separate entity.

3.2. Object-Centric Video Prediction

Video prediction is by definition a problem where the input and output are both in frame pixel space, however, for human eyes, it is not just about applying a transformation on an image. It is about understanding what is in the scene, what it is doing, and how this action or state evolves. Understanding scene dynamics, as well as object properties, is crucial for a correct prediction. A model that only sees pixels, without understanding for example the difference between a light and heavy object, will have a hard time understanding why objects that look similar behave very differently, depending on their material for example. Here comes object-centric video prediction into play. Object-centric representations can improve sample efficiency, robustness, generalization to new tasks, and interpretability of machine learning algorithms (Greff, Van Steenkiste, and Schmidhuber 2020).

Object-centric video prediction is a subset of video prediction approaches, where the prediction is not applied directly on the frames our their extracted features, but on a learned object representation. Object-centric representation learning is an active area of research, which will be discussed more in the next section. Here we talk more about video predictions model that utilize such representations.

One example we already discussed is the conditional video prediction model (Ye et al. 2019), where object representations are given with the image input in the form of feature description and location in frame. This initial object detection is done by the help of a pretrained ResNet-18 (He et al. 2016) CNN. Prediction is done in the entity representation space. To model object interaction, a Graph Neural Network architecture is used in the predictor. The graph structure is defined according to application, e.g. fully connected, or defined by human skeleton for human pose prediction. For modeling the stochasticity of future prediction, a global trajectory-level latent random variable is used. This variable is used to generate multiple plausible futures from just one frame as input.

Patch-based Object-centric Video Transformer (POVT) (Yan et al. 2022) also has object-centric representations that are generated from given boundary boxes. A Spatial Transformer (Jaderberg, Simonyan, Zisserman, et al. 2015) is used for extracting patches from each boundary box, and then they are down-sampled using a CNN for fixed size. Features are shifted from frames by one timestep, because they can only be extraced from an already generated frame. After preprocessing object representation, the approach uses a two stream transformer to model latent variables and object dynamics separately, then combines them using quadratic attention that spans both spaces.

For OCVT (Y.-F. Wu, Yoon, and Ahn 2021), the model needs to have explicit information about location and size in the representation. The input frame is divided into a grid of $H \times W$ size, and features are extracted from each cell and supplemented with this information. While this information is not needed in the dataset itself, the representation is kind of handcrafted to include them, so the representation is not fully learned.

As we see, in most object-centric video prediction approaches, the model need to be somehow initialized with object information to be able to represent the objects, or the representation needs to be human designed, to have a good enough representation to perform prediction on, thus requiring expensive human annotation. This is not the case in our approach, where the whole pipeline can be trained fully self-supervised. It does not need to be initialized by object information in form of extracted features by the aid of a pre-trained model, or through given information like boundary boxes or pixel locations.

3.3. Object-Centric Learning

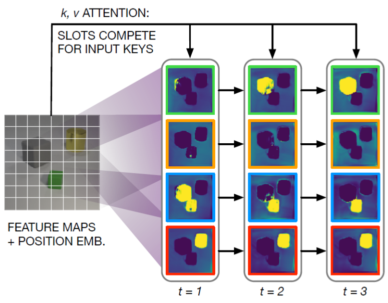
As seen from the examples above, there are various methods to represent objects in a scene. Some are handcrafted, and some used pre-trained models for object extraction and description. In this section, we discuss approaches that learn to recognize and describe objects in a self-supervised manner.

Self-supervised object-centric representation has been an active research area for both images (Burgess et al. 2019b; Greff, Kaufman, et al. 2019; Z. Lin, Y.-F. Wu, S. V. Peri, et al. 2020; Locatello et al. 2020; Villar-Corrales and Behnke 2021) and videos (Jiang et al. 2019; Kipf, Elsayed, et al. 2022; Watters et al. 2019; Weis et al. 2020). However, most of these approaches still only work on simpler synthetic datasets, and struggle with real life ones (Greff, Kaufman, et al. 2019; Harley et al. 2021).

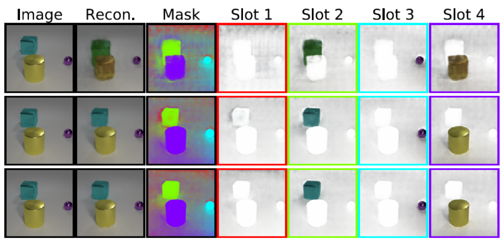
In the following we individually present the approaches relevant to this Thesis, which are Slot Attention and Slot Attention for Video. For them, while initialization with extra information can indeed be helpful, they also work well with random (learned) initializations, and this in fact is what we use in our experiments. They also provide flexibility in querying the objects and setting number of objects we want to see in a scene, to control the granularity of object detection, which are aspects that are helpful for video prediction. We can control the number of objects that we want to see in the scene and perform prediction on even with an already trained instance of a slot attention model. Nevertheless, the learned representation is strong enough for our application, as evident by applying prediction directly in object representation space, which will be showcased in detail over the upcoming chapters.

3.3.1. Slot Attention

In (Locatello et al. 2020), an architectural component called Slot Attention is introduced. It can be connected to the output of other componenets, e.g. CNNs, to produce a set of task-dependent representations, which are called slots. The



(a) Iterative process of slot attention.



(b) Reconstruction of slots.

Figure 3.2: Overview of the slot iteration process. **Left:** Iterative process of slot attention. Through the attention mechanism, the slots themselves act as queries that ask for more of the information they already include. **Right:** Reconstruction of slots after each iteration on objects from CLEVR dataset.

representations are extracted using an iterative procedure, that learns retrieval parameters generalizable to unseen compositions. This component can be utilized for multiple tasks on images, including object discovery and set prediction.

Iterative Slot Extraction

A representation of the iterative slot extraction process can be seen in figure 3.2a. A fixed number of slots is used in the model, which should be equal to the maximum number of object that can appear in an image, plus one slot for the background. Slots are randomly initialized from a learned distribution or completely at random. This randomness means that they have different initial distances to the representation of each of the objects. It also means that slots are unordered. No specific slots is always assigned to the same object, but this depends on the initialization.

The image is handled as a grid and feature maps of image parts are generated. A positional embedding is added to these features to keep spatial information. This input is used in an attention mechanism as keys and values, while the queries are the randomly initialized slots themselves. Intuitively, this means that at every iteration, each slot asks for more of the information it already contains. It also means that slots compete for the objects. Through iteration, each object converges to a slot. The background as well is assigned to one of the slots.

Image Reconstruction

This slot attention component is in itself just an encoder, that needs a decoder's feedback to learn what representation is needs in this slot space. Multiple tasks can be used for this, but for the purpose of this thesis, the object discovery task with image reconstruction is the most relevant.

As shown in figure 3.3, the image is reconstructed again from the information included in slots. By reconstructing the image from the slots, and using a reconstruction loss function, the model is forced to learn an object slot representation that is good enough for retrieving the image.

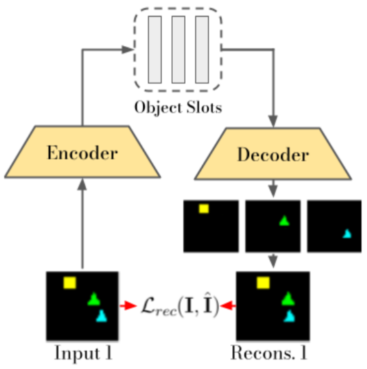


Figure 3.3: Architecture of the encoder decoder slot attention model. (Locatello et al. 2020).

The decoder works as follows: it generates a full image out of each slot, and a mask for each slot that specifies which areas are most relevant for the object in this slot. The masks are combined together by a applying softmax over the masks. As such, the slots are each competing for each pixel in the reconstruction. This allows for the reconstructions to be combined together in one final image. MSE is enough to act as a reconstruction loss function and proves sufficient for learning. Examples of reconstruction are shown in figure 3.2b.

Since both the decoder and the encoder are shared for all slots, and the whole difference between the slots is the initialization values, they can work with any number of slots needed, even if it differs from the number of objects in training dataset. For example, if we want to encode an image with the same structure, but we know it has more objects, we can increase number of slots at inference time to accommodate for this.

3.3.2. Slot Attention for Video

Slot Attention for Video (SAVi) (Kipf, Elsayed, et al. 2022) is the video extension of slot attention. It is applied directly on a sequence of related frames, and as such, leverages the information learned at one time step for information extraction in the next. The information flow is as follows:

Similar to slot attention, the first frame is passed to an encoder, which encodes the image information using feature maps and positional embedding. The initializer in SAVi is a dedicated component for the first frame, which either initializes the slots randomly or using information about objects in the scene, such as boundary boxes or pixel locations. A new component is introduced, which is described as the corrector. For the first frame, it compares the slot initialization, with the encoders output, and tries to adjust to contain better information. The output of the corrector is then passed to a decoder, whose type depends on the task. In (Kipf, Elsayed, et al. 2022), both tasks of predicting optical flow and image reconstruction are used to facilitate learning object representation. The output of the corrector is also passed to a predictor, which is a transformer module designed for predicting the upcoming slots in the next time step. For the following time steps, the output of the predictor is used instead of the initializer. The predictors are corrected by the corrector, and the recursive process of predicting and correcting goes one over the video. Whole overview of the process can be seen in figure 3.4.

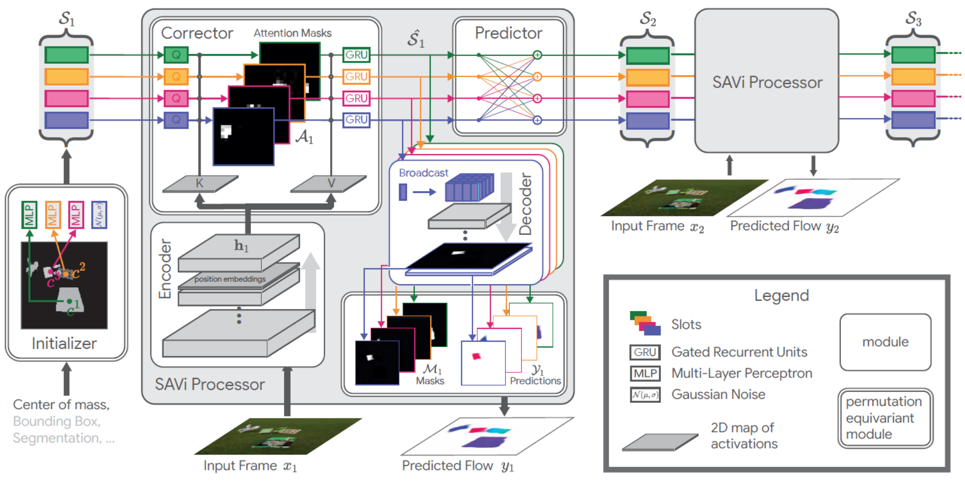


Figure 3.4: Overview of SAVi architecture (Kipf, Elsayed, et al. 2022).

4. Methodology

After reviewing the literature and the current state-of-the-art in both object-centric learning and video prediction, we seek an approach that provides object representation learning that is both interpretable as well as as flexible as possible. Ideally, we want an approach that is fully self-supervised, without the need for any hints about objects, e.g. bounding boxes or location, and with the ability to adjust the number of objects and how granular they are divided, or on whether we want larger objects, or we want them divided into more detailed subparts. We also want a representation that is good enough for the scene to be reconstructed back from it with enough quality, and that contains enough information to be used in prediction directly in this object representation space. As such, we want a prediction pipeline that focuses equally on both the end reconstruction result, as well as having meaningful predictions in the object representation space.

Our proposal is to use the slot attention approaches discussed in the previous section, whether the Slot Attention Model, or SAVi, for the object representation learning, and try to combine them with predictors suited for the learned represenation, and that generate results that can be reconstructed back by the same models decoder to form the predicted scene. We experiment with different approaches for prediction, including RNN and transformer-based approaches, to try to find the best one for performing prediction in this object slot space, and how each of them handles the task and affects the results.

This chapter is divided into the two main two topics of the thesis. The first section focuses on the object representation learning using Slot Attention and SAVi, what steps were taken there, and what final setup we arrived to. The second section focuses on the prediction. We introduce our used predictors, including two new transformer-based predictors designed for object-centric prediction, and discuss how each of them works and what their advantages and disadvantages are. We finish the chapter by discussing the complete training pipeline and any details relevant to how the predictions are performed.

4.1. Architecture overview

The success of the slot attention models in capturing object information encourages using this information in different tasks. This thesis investigates the possibility of performing prediction tasks on this slot representation. To validate this prediction we evaluate the reconstruction of the predicted slots and compare it with ground truth frames. The overview of architecture is shown in figure 4.1.

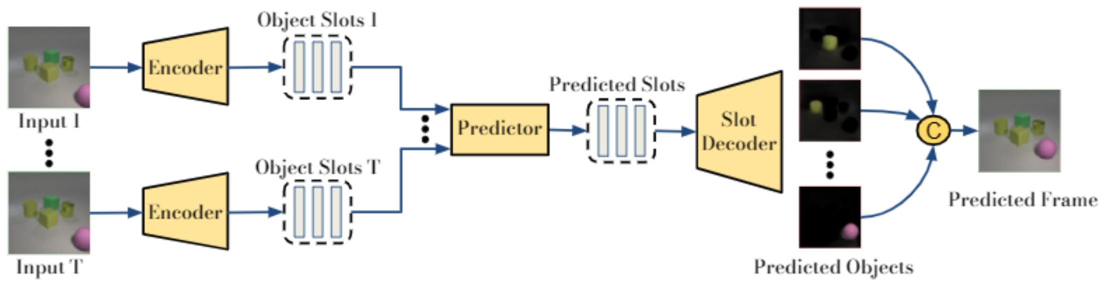


Figure 4.1: Overview of our proposed approach.

For a predefined number of context frames T, the frames are encoded into their object slot representations of N_s slots each of dimension D_s. A tensor of dimension $T \times N_s \times D_s$ is passed to a predictor model, that generates a tensor of dimension $N_s \times D_s$ describing the predicted frames of time step $T + 1$.

The ground truth target frame is also encoded into the slot space, and the model tries to match the ground truth in both slot space as well as frame space, using two equally-weighted loss functions. We use pre-trained instances of Slot Attention and SAVi to preform the encoding and decoding of the scene into the object-centric represention (slot representation). They are frozen through training, and are not affected by the learning process. It is the predictors task to learn to deal with the representation generated by them.

4.2. Slot Extraction

4.2.1. Slot Attention and Slot Alignment

The slot attention model provides the needed encoder component in the above architecture, so we needed to perform experiments on the used datasets to find the best configuration for training Slot Attention, since the results from the encoding are of vital importance for the rest of the pipeline. We started with the simplest data, which is VMDS, to run quick experiments and notice problems early. Slot

Attention is designed to work on images not videos, so the dataset was fed as individual frames to the model.

As mentioned, objects in a frame build an unordered set. The slots generated are also unordered, and are assigned to each object randomly, depending on the initialization. As such, when encoding two consecutive frames, slots need to be matched according to the object they contain. We theorize that the slot representation of the object is well-defined enough to recognize objects and calculate similarities within it. After calculating similarity scores, a matching algorithm needs to be used to align the slots and track the objects over time. Namely, we employ the Hungarian algorithm (Kuhn 1955) to find the best matching between the object slots in consecutive video frames.

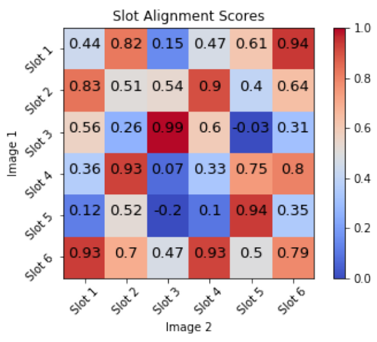
4.2.2. Initial Results on VMDS and Slot Alignment

After obtaining a successfully trained instance of Slot Attention on VMDS, we used it for working on slot alignment. For a similarity measure we used both L2-distance, as well as cosine similarity:

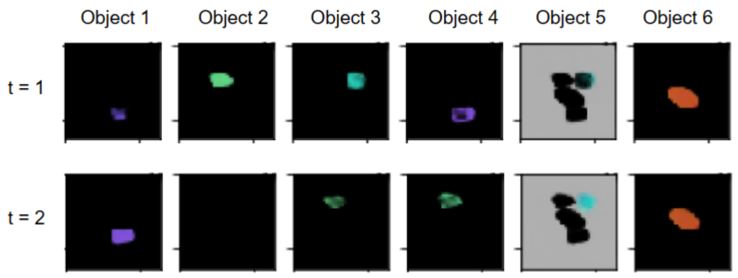
$$S_c(s_1, s_2) = \frac{s_1^T s_t}{\|s_1\| \cdot \|s_2\|} \tag{4.1}$$

Using these similarity measures, we generated distance matrices between slots of time step t and $t+1$ for each $t \in T$, and then apply the Hungarian algorithm for finding the best matches. Example of scores generated for 6 slots at time steps 1 and 2 can be seen in figure 4.2a. The matching is done step by step in time, where each slot was matched with the best match in the immediate past to form a linked list of slots belonging to the same object.

We have found through qualitative analysis that the alignment performance depends heavily on having the right number of slots. In training, we initialize the model with the highest number of objects expected in a scene in the dataset. If the number of slots is higher than number of objects, the performance of alignment goes down. If we extract the number of objects from the test data, and initialize the model with an equal number of slots plus one for the background, we often get good results, like in figure 4.3b. If we just use number we used in training, we get worse results as shown in 4.3a. In both cases however, the alignment is still not perfect and we argue that the learned object representations are likely not good enough for training our object-centric predictor modules.



(a) Similarity scores matrix.



(b) Object matching between slots.

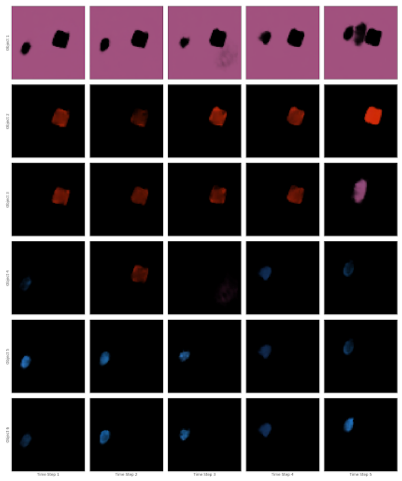
Figure 4.2: Slot alignment using cosine similarity and Hungarian method. **Left:** A similarity score matrix generated by cosine similarity. **Right:** Object matching between slots depending on the scores shown in figure 4.2a. First row is the individual slot reconstruction at time step 1, and second row are the corresponding matches found from time step 2.

4.2.3. Switch to SAVi

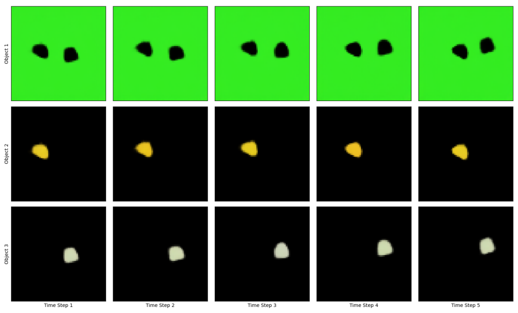
Trying to match the slots proved to be too complicated and error prone, so we swtiched the SAVi model. SAVi is designed to work on videos directly. The slots are randomly initialized only for the first frame, then passed as a seed to the slot iterations of the next frame and so on. This works because there are minor changes between consecutive frames and usually contain the same objects. It also means that the slots will nearly always keep their order, because the object that it is most similar to a certain object is itself, even if it moved a little or its shape slightly shifted. The query in slot attention will always ask for information about the object that was in the slot in time step before. An example is shown if figure 4.4, where a perfect aligment is automatically found by the model.

4.3. Predictors

The main contribution of this thesis is the investigation of various predictors and their inductive biases when performing the combined task of predicting future slots and video frames. We investigate four different predictors: a traditional LSTM model, a regular vanilla transformer encoder, and introduce two new types of transformer encoders for the specific task of object-centric video prediction.



(a) Alignment with the fixed number of slots as in training.



(b) Alignment with the right number of slots needed for scene.

Figure 4.3: Example of slot alignment when initializing the model with same number of slots as in training vs. when providing correct number of objects.

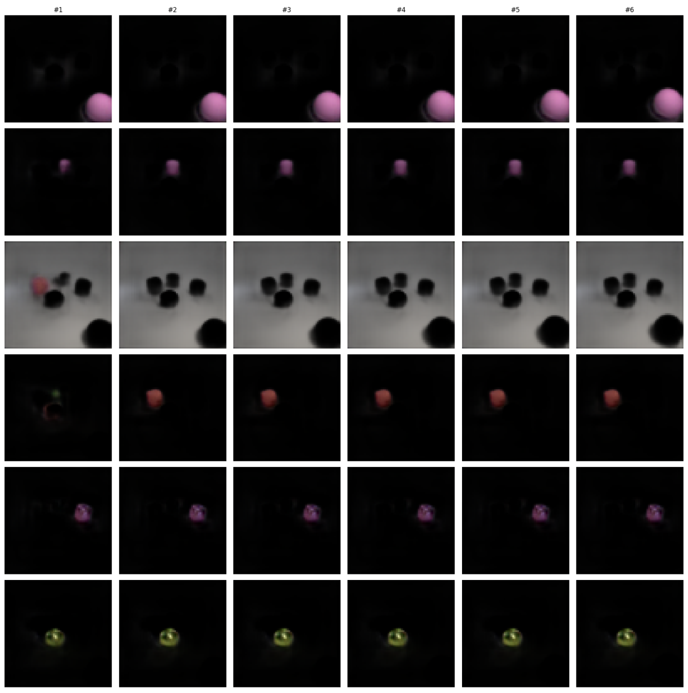


Figure 4.4: Readily aligned slots generated by SAVi.

4.3.1. LSTM

Given a sequence of slots belonging to the same object, an LSTM model can learn to recursively predict the next slots. The LSTM is fed the slots representing a single object extracted by the encoder from the context frames, and tries to predict the next slot. During training, it can be fed back its own prediction, or teacher forcing, which is the process of feeding the model the ground truth instead of its own output, can be used to make sure the input is always correct. The process is repeated for all slots, and at the end the predicted slots are collected according the time step they belong to, and slots of each time step are decoded together to reconstruct the predicted frame.

The advantage of such model is that it is not limited by a chosen number of context frames or predicted frames. It can iterate as long as context frames are available. The disadvantages include the usual RNN disadvantages of limited information. The LSTM tries to summarize all context frames in its internal state, so it does not have access to the whole information from the past. Another disadvantage is that only looks back in the temporal dimension, but cannot perceive the spatial dimension, and the interaction between objects. Each slot is predicted on its own without knowledge of other slots.

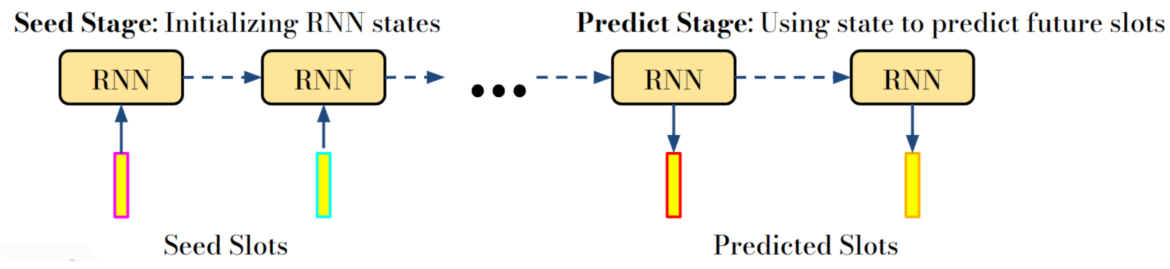

Seed Stage: Initializing RNN states
Predict Stage: Using state to predict future slots
RNN
RNN
RNN
RNN
Seed Slots
Predicted Slots

Figure 4.5: LSTM based prediction of slots.

4.3.2. Vanilla Transformer

Unlike RNNS, transformers are able to take a look at the whole input sequence at once and choose from it the most important information using attention. The vanilla transformer predictor that we use takes $T \cdot N_s$ slots of all seed time steps. It uses self-attention over all of them and is able to choose the relevant information for the prediction. Architecture is shown in figure 4.6.

Transformers do not have notion of order of the input, which is why the model needs positional encoding. Since objects are unordered anyway, we just need positional encoding along the time dimension. We apply along it the encoding mentioned in equation 2.2.

Complexity of this predictor is $O((T \cdot N_s)^2)$, and hence, unlike the LSTM predictor, it is able to model interactions between objects and can consider what is happening for one object when predicting what will happen for another. This

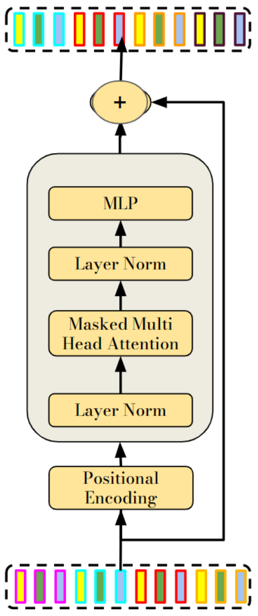


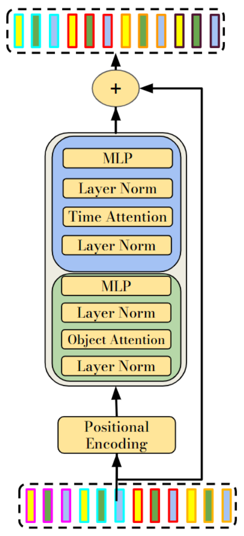
Figure 4.6: Vanilla transformer prediction of slots. The models gets all slots from all context frames as input, and outputs them shifted by one step such that the last set of slots are the predicted slots.

would enable it to make predictions during events such as collisions, occlusions, and momentum transfers.

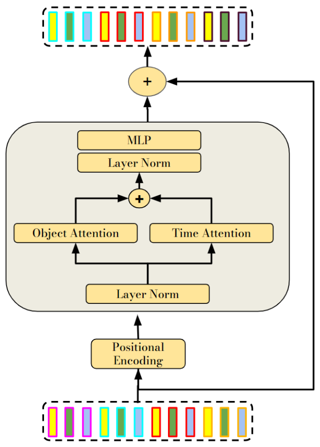
This predictor, however, is not very structured, and as such can be hard to interpret. Here the model is tasked with predicting the future given all available information about everything that happened to all objects. No structure is given to help the model understand interactions and dynamics. The model can still learn to generate meaningful prediction, because all the information it needs is there, but for human it might be hard to interpret the results. For example, it might no be clear what does it mean if the model attends to an object's past to predict the future of another object. During evaluation, we see some examples of how this high complexity affects the model and the self-attention.

4.3.3. Sequential Object-Centric Video Transformer

We propose a novel transformer-based predictor architecture that aims to reduce the complexity of the regular transformer discussed above, by separating the processing of temporal and spatial dimensions. All slots of all time steps are given together to the module as input, but it passes them through two consecutive phases. First, it employs an object attention block, which attends for each time step only to the object states at this time. The output of this block is an encoding



(a) Sequential Object-Centric Video Transformer.



(b) Parallel Object-Centric Video Transformer.

Figure 4.7: Our two novel object-centric video transformers. **Left:** The sequential object-centric video transformer consists of two consecutive phases, where the model attends first to the spatial dimension (other objects in the scene at the same time step), then to the temporal dimension (the states of the object itself in the past). **Right:** The parallel object-centric video transformer consists of two concurrent phases, where the model attends to the spatial dimension and to the temporal dimension in parallel, and processes the results together.

for each slot that describes it and its surroundings. It is passed to the next module, the time attention block, that for each slot attends to its representations along time. These representations now include encoding of the surroundings, so the transformer is able to observe how the slot and its relevant surroundings change according to time. Architecture is shown in figure 4.7a.

This model also takes the input with positional encoding along the time dimensional. For controlling the attention, we generate a mask for each block, that defines the elements allowed to be attended to for each slot. Unlike the vanilla transformer, the complexity here is linear in $O(T^2 + N_s^2)$, instead of $O((T \cdot N_s)^2)$.

This separation of dimensions in the attention blocks makes it easier to interpret the process of prediction inside the object-centric transformer. Each block answers a direct and intuitive question. The object attention block answers the question of how the current object is affected by its surrounding objects, i.e., it answers the question of object interactions. The time attention block answers the question of how the object's past influences its future, i.e., it answers the question of object dynamics. Both questions together are highly relevant to the prediction task, as well as understandable and intuitive ways for the human mind to view the prediction task.

4.3.4. Parallel Object-Centric Video Transformer

Our second proposal goes a step further in the direction of efficiency. Instead of consecutive processing of the two dimension, the blocks are merged into two parallel paths and the outputs are added after the attention mechanisms. The normalizing layers and MLP are shared between them. Architecture is shown in figure 4.7b.

Parameter sharing means less parameters to train, so the model is more efficient to train. However, since the attentions take separate paths and the output of one does not affect the other in any direct way, the model cannot check relations between slots of different at different times. Whether this poses a problem or not is a question that will be answered with the experiments.

4.3.5. Attention Masks

For all three transformers, attention masks are needed to limit the attention span to only the past time-steps, because we do not want the model to just copy for a certain slot its future as the prediction. Moreover, masks are used along the different dimensions to control the kind of attention each block has in the object-centric transformers.

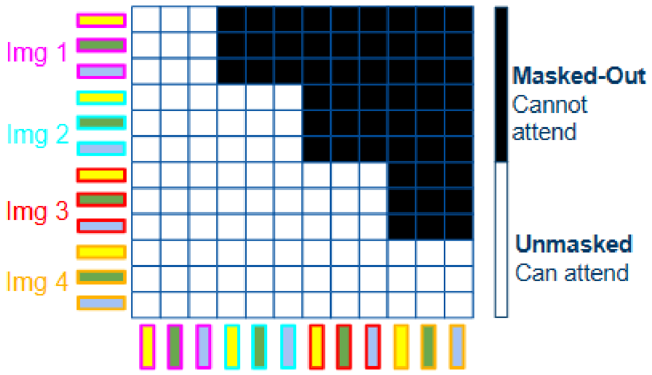


Figure 4.8: Attention mask for vanilla transformer for 4 frames each encoded into 3 slots. The model attends to all slots in all time steps.

Object attention and mask attention differ in the areas they span in the attention map. Object attention describes an isolated block of the all slots at the a certain

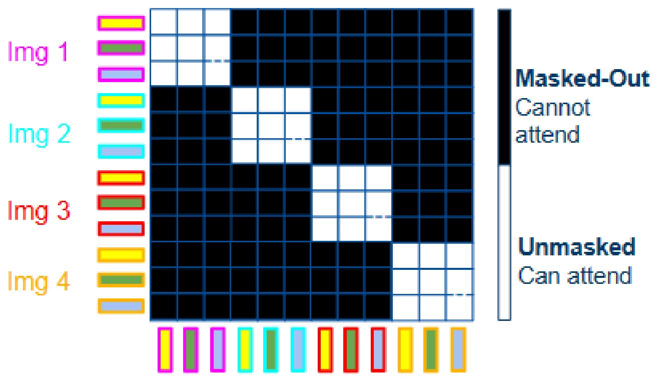


Figure 4.9: Attention mask for object attention in object-centric transformers for 4 frames each encoded into 3 slots. For each slot the model can only attend to other slots at the same time step (same frame).

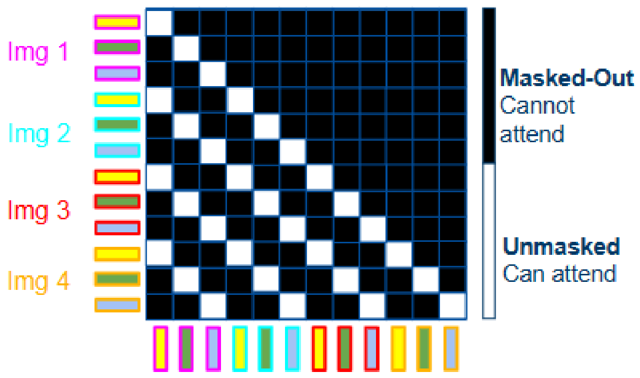
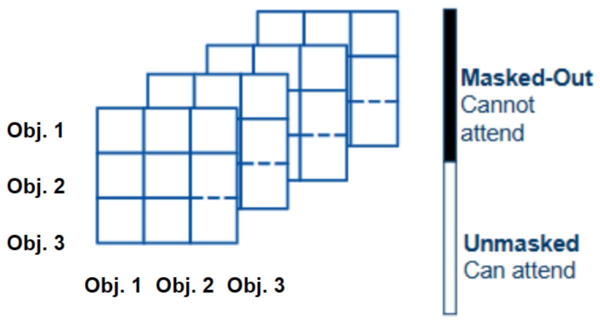
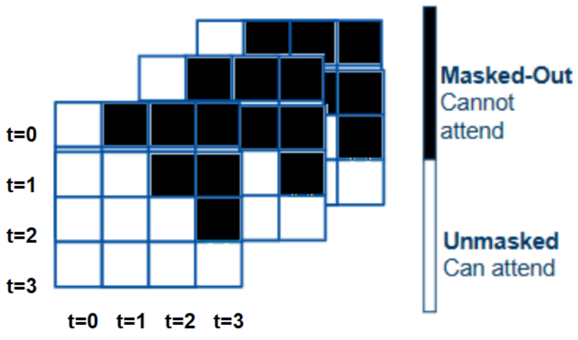


Figure 4.10: Attention mask for time attention in object-centric transformers for 4 frames each encoded into 3 slots. For each slot the model can only attend to the same slot at the current and past time steps (current and previous frames).



(a) Batch-wise implementation of object attention.



(b) Batch-wise implementation of time attention.

Figure 4.11: Implementation of object and time attention for $T = 4$ frames and number of slots $N_s = 3$. **Left:** Attention masks for object attention in the batch-wise implementation collecting time steps together. **Right:** Attention masks for time attention in the batch-wise implementation collecting slots together for time attention.

time step. It has no dependency on whatever came before or whatever came after. This is why in figure 4.9 we see it visualized as square blocks on the diagonal, such that no two blocks ever overlap in the same row or the same column. On the other hand, time attention focuses on one thing, namely one certain slot, doing this repeatedly every time step. This is why it appears in figure 4.10 as a repeated function with a period equal to the number of slots, with each line shifted by one from the line before.

While object and time attention masks can conceptually be visualised by figures 4.9 and 4.10, in the actual implementation, we divide the slots into batches along the dimension on which we want the attention to take place. For object attention, this simply means we have batches of slots exisiting in each time step. The resulting attention map, as shown in figure 4.11a, is a three dimensional tensor of shape $T \times N_s \times N_s$, describing at each time t how much each slot attends to other slots of objects in the frame. For time attention, the batches are along the number of slots, and for each slot the batch consists of its versions along time. Since we want to model to only use the past for prediction not the future, the attention maps have a lower-triangular shape, depicted in figure 4.10, which only allows attention on previous indices. The resulting attention map is a three dimensional tensor of shape $N_s \times T \times T$. Attention masks of this batch-wise implementation are shown in figure 4.11b.

In figure 4.12, we see an example of how the time attention would work at prediction. The attention will usually be the strongest at the nearer time steps, and decreases with distance in the past. For more static objects, the attention can be more divided over the time steps.

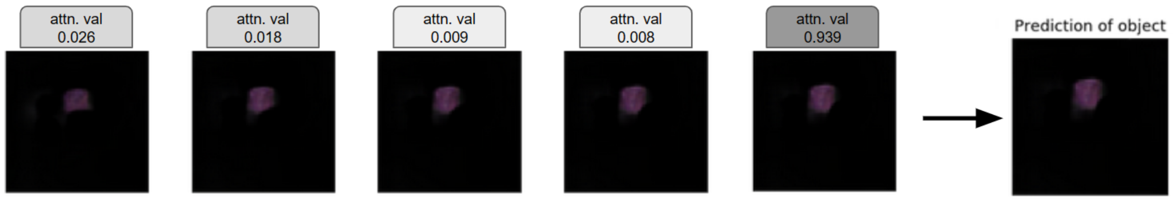


Figure 4.12: Attention values for time attention.

4.3.6. Building Choices for the Predictors

All predictor modules, LSTMs and transformers, can be stacked to increase the capacity of the model. We test different configurations of multiple LSTM cells together, and multiple transformer blocks together. We also test the effect of having residual connections between these blocks on the model performance.

Other choices for predictor training include teacher forcing and skipping first slots. When predicting more the one frame in the future, we have a choice to used already predicted frames as inputs, or the ground truth frames, also known as teacher forcing. Teacher forcing can facilitate model training by providing it with the correct input to be able to learn the relation between the input and output better. However, it can bias the model and make it unable to handle its own imperfect predictions, which is something that is needed when it is fed its output again in inference time. We test this in our ablation studies to see what works best.

As mentioned about SAVi, the first time step receives randomly initialized slots, while the following slots used their predecessors as a seed that is corrected to the new input. As a result, the performance of the SAVi model is always worse at the first time step. This might affect the predictor training, if the slot description is not of enough quality. A choice of discarding the slots of the first time step can be made, and we also include it in our ablation studies.

4.3.7. Loss Function

Our choice for loss function is a combined loss of the slot prediction and reconstructed frame prediction. We apply MSE on both of them, and give them equal weights. We want the model to be focus equally on predicting the same slot representation that is generated by SAVi, as well as giving a resulting reconstruction that is close to what happens. This results in the loss function in equation 4.2, where the P is for slot space outputs, and I is for frame space outputs. The loss computation is also visualized in figure 4.13.

$$\mathcal{L} = \mathcal{L}_{rec}(I, \hat{I}) + \mathcal{L}_{pred}(P, \hat{P}) \tag{4.2}$$

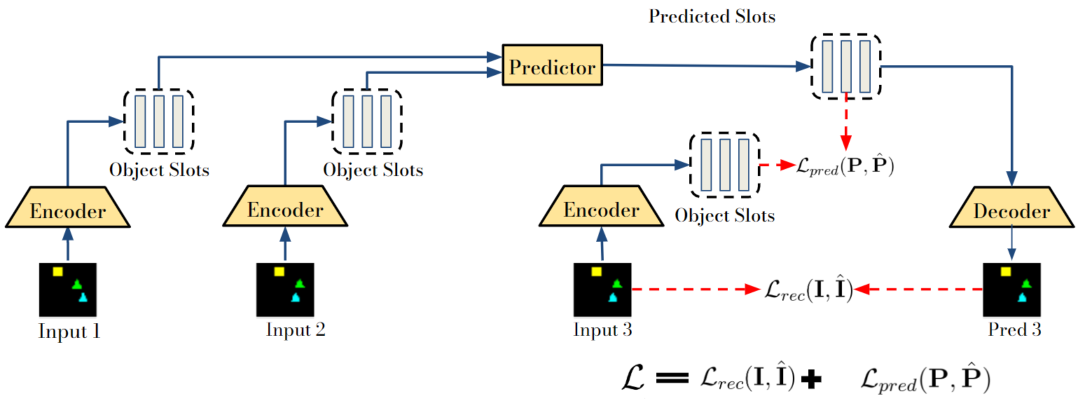

Predicted Slots
Predictor
Object Slots
Object Slots
Object Slots
Encoder
Encoder
Encoder
Decoder
$\mathcal{L}_{pred}(\mathbf{P}, \hat{\mathbf{P}})$
$\mathcal{L}_{rec}(\mathbf{I}, \hat{\mathbf{I}})$
Input 1
Input 2
Input 3
Pred 3
$\mathcal{L} = \mathcal{L}_{rec}(\mathbf{I}, \hat{\mathbf{I}}) + \mathcal{L}_{pred}(\mathbf{P}, \hat{\mathbf{P}})$

Figure 4.13: Loss function computation using predictions in slot and frame spaces.

5. Evaluation and Results

5.1. Datasets

A number of datasets have been collected and labelled for the different tasks of video analysis and prediction, varying in length, complexity, object interactions, dimensionality (two or three dimensional objects), and being natural or synthetic. For video prediction as a self-supervised learning problem, any video dataset can be used. However, some datasets are much more challenging than others. The following are the datasets that are relevant to our experiments.

5.1.1. VMDS

Video Multi-dSprites (VMDS) (Burgess et al. 2019a) is a dataset consisting of simple two dimensional geometric shapes of consistent colouring and colourful backgrounds. Each frame contains a maximum of 5 objects. There is no interaction between objects, however, they can come in front of each other causing occlusions. Frame size in this dataset is 64×64.

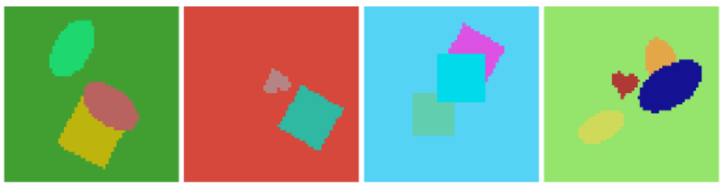


Figure 5.1: Example of frames in multi-dSprites and VMDS datasets (Burgess et al. 2019a).

5.1.2. Obj3D

Obj3D (Z. Lin, Y.-F. Wu, S. Peri, et al. 2020b) is a video extension of the CLEVR dataset (J. Johnson et al. 2017). It includes colourful three dimensional objects on a gray plane. The complexities in the dataset include lightening direction, occlusions in a 3D environment, collisions, and acceleration. While CLEVR as an image dataset was built to include a lot of semantic information for the tasks of visual reasoning and question answering, a lot of this information is lost in the extension to videos. As such, this dataset does not provided labelled masks for the frames. Frame size in this dataset is 64 × 64.

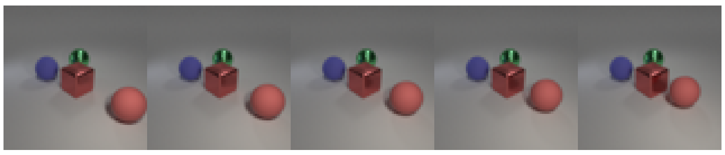


Figure 5.2: Example of a 5-frame sequence in Obj3D (Z. Lin, Y.-F. Wu, S. Peri, et al. 2020b). The ball is moving towards the objects and will collide with them bringing them to motion.

5.1.3. SynPick

SynPick (Periyasamy, Schwarz, and Behnke 2021) is another synthetic dataset, designed for the task of bin picking. The data consists of videos of a mechanical arm picking up objects inside a box. This dataset is a challenging dataset, because: it shows the movement of an active agent that makes decisions to achieve its goal, and it has a high number of objects (up to 21 objects). The actions of moving and picking introduce variety in scene dynamics, and objects interact together in different ways, such as collision and occlusions. The dataset also offers different lighting options and three different views of the box. The dataset has automatically generated object annotation. It can be considered as a tough test for video prediction models. Frame size in this dataset is originally Full HD 1920 × 1080. For our experiments we resize it to 64 × 112.

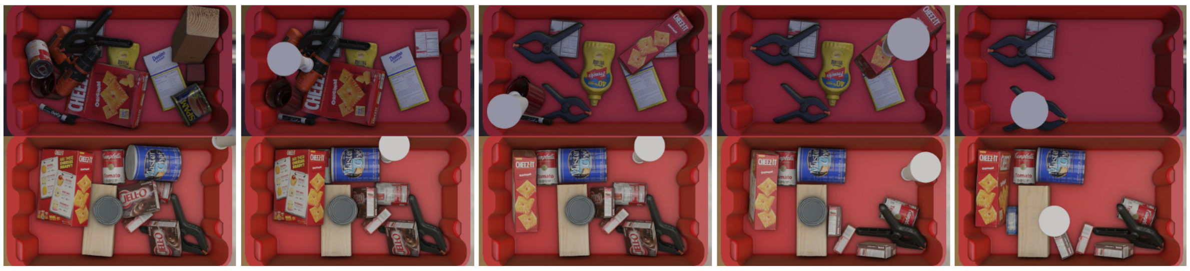


Figure 5.3: Example of scene in the SynPick dataset(Periyasamy, Schwarz, and Behnke 2021). The robot gripper is moving and picking up different objects, in different light settings. As it moves, it covers and uncovers objects in the scene.

5.1.4. Sketchy

Sketchy dataset (Cabi et al. 2019) is a real video dataset, designed for the task of picking and stacking objects. The data consists of videos of a mechanical gripper picking up objects on a flat surface and moving them. It is another challenging dataset, because it also shows the movement of an active agent that makes decisions to achieve its goal, however it has fewer objects (maximum of three objects per scene, in addition to the gripper) and the viewing angle is similar to Obj3D dataset. Difficulty-wise for the model, it might be to be more complicated to learn than Obj3D, but simpler that SynPick, due to the viewing angle and the fewer number of objects. However, the gripper in Sketchy has a more complex structure and movement. Experiments should say more about how the datasets have different demands from the model. Frame size in this dataset is 600 × 960 but we resize it in our experiments for 80 × 120.

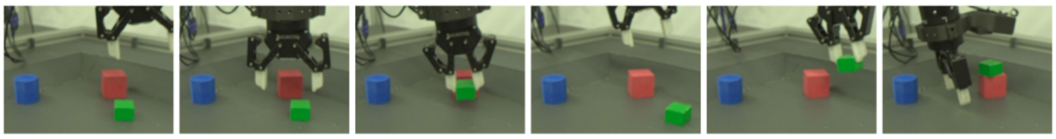


Figure 5.4: Example of scene in the Sketchy dataset (Cabi et al. 2019).

5.2. Video Prediction Metrics

Video prediction metrics try to compare the quality of prediction by comparing the similarity of a predicted frame to a ground truth frame, and averaging over length of prediction. We use the following functions in our experiments. MSE is mainly used as the loss function, but not for evaluation, while we report the rest of these metrics as our evaluation metrics.

MSE

Mean squared error is a common learning metric for regression tasks. It is used as a loss function that is to be minimized, so as a metric the less it is the better, down to a perfect zero.

$$MSE = \frac{1}{TN}\Sigma_{t=1}^{T}\Sigma_{i=1}^{n}(Y_{it} - \hat{Y}_{it})^2 \tag{5.1}$$

PSNR

Peak signal-to-noise ratio (PSNR) is the ratio between the maximum power of an image signal and the power of noise corrupting it. As such, it is a metric that is desired to be as high as possible (with no theoretical maximum if there is no error). It can be defined by MSE using:

$$PSNR = 10 \cdot \log_{10}(\frac{MAX_I^2}{MSE}) \tag{5.2}$$

LPIPS

Learned Perceptual Image Patch Similarity (LPIPS) (R. Zhang et al. 2018) tries to capture image similarity in a way that coincides with human judgment. In order to do this, MSE is not directly calculated over the exact output and ground truth. Instead, both of them are passed through a pre-trained CNN, e.g. VGG, that generates meaningful features maps for both of them. The distance is calculated between these two features maps, since they should be more focused on values and details relevant to visual tasks. As with any distance metric, it is desired to be minimized in our case.

SSIM

Structural similarity index measure (SSIM) (Z. Wang et al. 2004) is a measure of image similarity that tries to capture the overall structure similarity instead of the per pixel error. It is defined by:

$$SSIM(X,Y) = \frac{(2\mu_Y \mu_Y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c1)(\sigma_x^2 + \sigma_y^2 + c_2)} \tag{5.3}$$

where X and Y are the two images we measure similarities for, μ_X and μ_Y are their means, σ_x,σ_y and σ_{xy} are their variances and co-variances. c_1 and c_2 are two stabilizing variables for the division. This formula results in values between 0 and 1, where 1 indicates the two images are identical.

5.3. Experimental Setup

We used VMDS only for preliminary experiments, but focused our work on the three more complicated experiments of Obj3d, Sketchy, and SynPick. We follow the guides of the original SAVi paper in training where-ever possible on the Obj3D and Sketchy datasets. After multiple experiments, configurations that reached best results are listed in table 5.1.

Table 5.1: Training Setup For SAVi According to Dataset

	Obj3D	Sketchy	SynPick
No. Slots	6	11	14
Slot Dimension	128	128	256
Slot Attention Iters.	3	3	3
Input Frame Size	64×64	80×120	64×112
Slot Initializer	Learned Random	Learned Random	Boundary Boxes
Encoder	Conv. Encoder	Conv. Encoder	ResNet34
No. Frames in Input	10	5	10
Batch Size	64	32	16
No. Epochs	2000	240	100
Starting LR	1e-4	1e-4	1e-4
LR Scheduler	cosine annealing	cosine annealing	cosine annealing
Optimizer	Adam	Adam	Adam
LR warm-up	✓	✓	✓

Note that these configuration are not exhaustively optimized to reach the best possible values. We trained the SAVi instances till reaching reasonably good results without introducing extra requirements and complexities. For example, the learned random initialization was good enough for Obj3D and Sketchy, however, for SynPick we needed to introduce boundary box initalization. Same thing with the simple convolutional encoder and the ResNet34 used as encoder for SynPick.

For prediction training, the configurations are similarly summerized in table 5.2.

Table 5.2: Training Setup For Predictors According to Dataset

	Obj3D	Sketchy	SynPick
No. of context frames	5	5	5
No. of predicted frames	5	5	5
LSTM Hidden Dimension	128	128	256
Trans. Token Dimension	128	128	256
Trans. Hidden Dimension	256	256	512
Trans. No. of Heads	4	4	4
Batch Size	64	32	64
No. Epochs	1500	60	500
LR	1e-4	1e-4	1e-4
LR Scheduler	constant	constant	constant
Optimizer	Adam	Adam	Adam
LR warm-up	✓	✓	✓

5.4. Evaluating SAVi on Used Datasets

Our training attempts for training SAVi on the three datasets achieve different degrees of success, with the order expected from the complexity of the datasets. We are able to achieve good results on Obj3d, and satisfactory results on Sketchy and SynPick, enough to test the predictors. The reconstruction results can be seen in table 5.3.

It is worth mentioning that we would expect to achieve better results if we were able to try experimenting more with model parameters, capacity, and structure, however, due to the time constraints, and since one training running on the more complicated datasets needs around one week, we were not be able to do so. As a result, we focus mainly on Obj3D dataset, and also report results on Sketchy and SynPick.

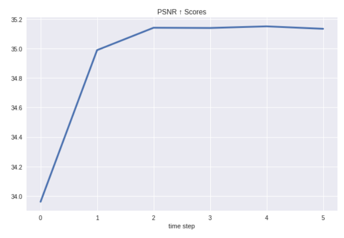
Table 5.3: Evaluation of SAVi instances we trained on used datasets (averaged over sequence length used per dataset).

	PSNR↑	SSIM↑	LPIPS↓
Obj3d	34.919	0.953	0.020
Sketchy	28.705	0.912	0.081
SynPick	26.171	0.728	0.192

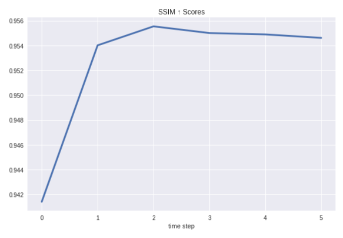
When evaluating SAVi results frame-wise, we see that the metrics for the first frame are always below average, and get better over the next two frames before converging. This is expected, because the first frame is initialized randomly, while the second frame starts from where the first frame ended, so it gets a better result, and the third starts from where the second ended, so it gets an even better result. Then the metrics stays consistent, because the slot initialization is consistent in quality. These numbers on the Obj3D datasets can be seen in table 5.4 and the trends in them are visualized in figure 5.5c. The findings are the same over all three metrics. Qualitative results shown later confirm these findings.

Table 5.4: Average metric values with respect to time for SAVi trained on Obj3D.

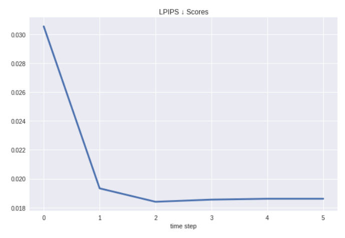
	Frame 1	Frame 2	Frame 3	Frame 4	Frame 5	Frame 6
PSNR↑	33.9601	34.98925	35.14135	35.13971	35.15079	35.13416
SSIM↑	0.9414	0.95401	0.95554	0.955	0.95489	0.95461
LPIPS↓	0.03055	0.01933	0.0184	0.01855	0.01861	0.01861



(a) PSNR scores.



(b) SSIM scores.



(c) LPIPS scores.

Figure 5.5: Plots of evaluation metrics according to time step for SAVi model trained on Obj3D. A trend of improving performance as the model sees more similar frames is observable.

In figure 5.6a, we observe visually how the reconstructions of SAVi compare to the input on the Obj3D. We see the input, output, and difference between them. We notice that the first frame indeed sometimes has some artifacts that get fixed over the following frames. Object decomposition results can be found in Appendix A.

We repeat the same for Sketchy and SynPick in figures 5.6b and 5.6c. The results are not as good as the ones on Obj3D, as to be expected from the numbers. The objects seem to be smoothed, which might be an indication that the models needs higher capacity to capture a higher resolution reconstruction, but this needs more investigation. This smoothing is the reason the difference images mostly correspond to object edges.

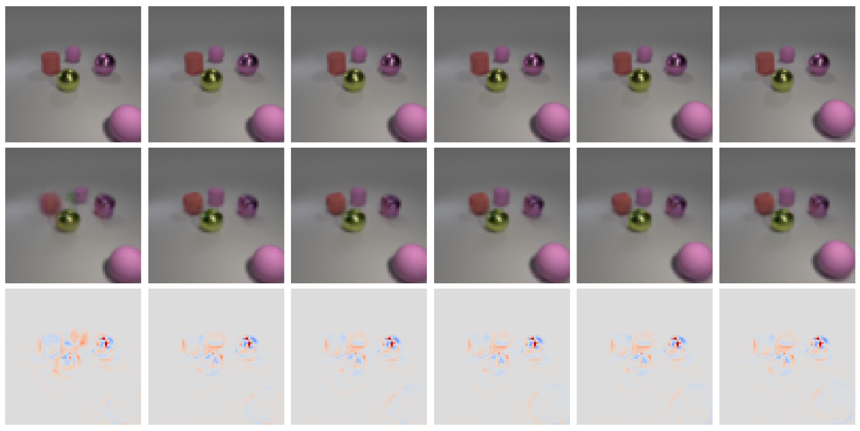
5.5. Ablation Studies of Predictors

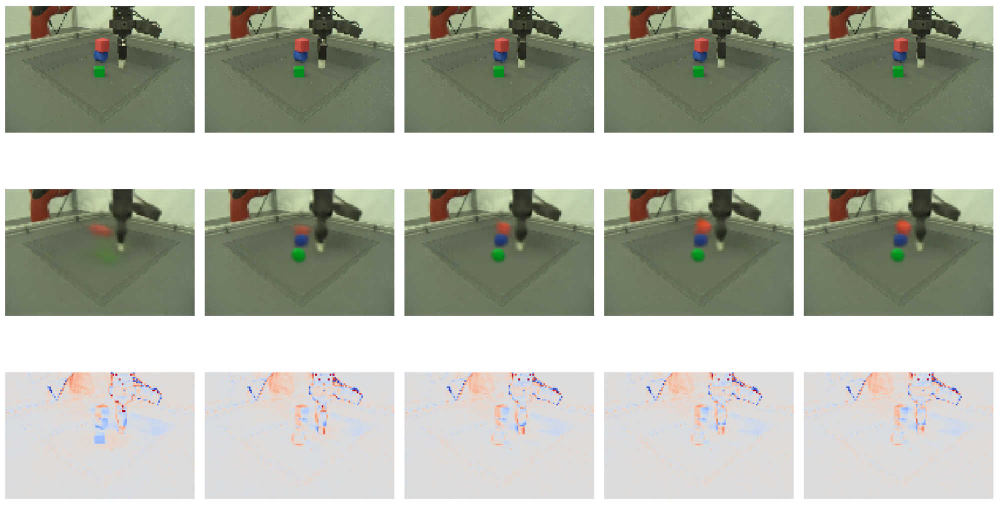
To better understand the model and training requirements to perform object-centric video prediction, we perform several ablation studies using our predictors using the trained instance of SAVi on Obj3d, since it achieved the best results.

Using the vanilla transformer, we test the effect of the following choices: using teacher forcing, skipping the first slot time-wise (the one that is randomly initialized in SAVi), and having residual connections bridging the predictor modules. The results of our ablation studies are listed in table 5.5.

For teacher forcing, we find that it sometimes helps the model achieve better results on the shortest prediction horizon of 5 frames, but without it the model learns to handle its own imperfect predictions better, and as such performs better for the longer prediction horizons for 15 and 30 frames. For this reason, we choose to train without teacher forcing for the rest of the experiments.

Skipping the first time step does not seem to help the model predict better


(a) SAVi reconstruction on Obj3D.


(b) SAVi reconstruction on Sketchy.

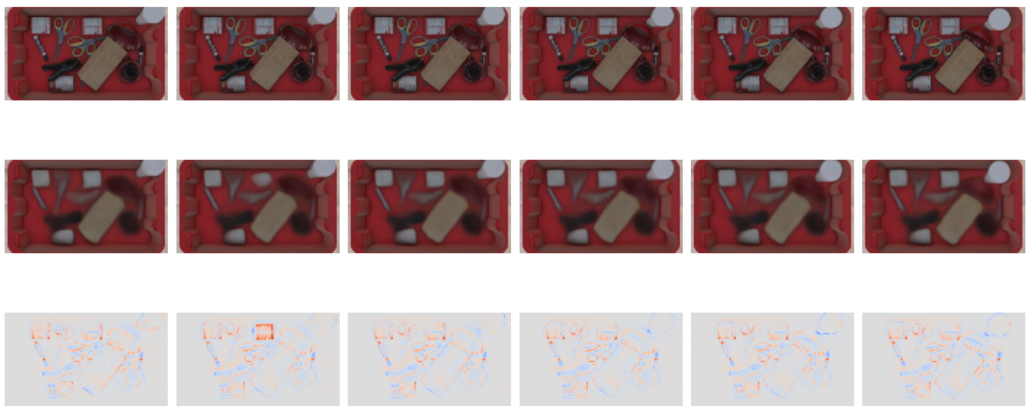

(c) SAVi reconstruction on SynPick.

Figure 5.6: Examples of reconstruction of SAVi using the decoded slots on the three datasets, and how they compare to the input. In each example, we have row 1: input, row 2: output, row 3: difference between them.

frames, often leading to the same LPIPS and SSIM scores, even though it is usually the lowest quality reconstruction for SAVi. This might be due to decreasing the number of available context frames by discarding this time step. We choose to train later models without skipping the first time step.

For the number of layers, we found that training the model with 2 transformer blocks was the optimal spot. Just one layer was not enough capacity for the model, and four and six layers were causing overfitting. We decide to stick with 2 layers for the rest of the experiments.

Finally, we test the model using residual connections between each transformer block. We find that having residual connection had the greatest effect on the model performance. A reason for this could be that the model keeps track of the original structure of the slots and the output of transformer encoder going to the next encoder is used as supplementary information and not a replacement. We use residual connections for the rest of the experiments.

After finding this large effect of residual connections, we repeat the experiments again with residual connections, to have accurate representation of their effects. The final results can be seen in table 5.5. The affect the residual connections have is very prominent and we deduce that they act against the overfitting effect of adding more layers, and that now the 4-layer transformer is the best performing version.

In summary, our ablation studies find that for transformer predictors, the best setup consists of: using no teacher forcing, using residual connections, and stacking 4 predictor layers together. For skipping the first time step, no significant difference is noticed, especially after adding the residual connections, and it can be done either way. We apply these findings when training the object-centric transformers.

We then move on to performing ablation studies to try to find best configurations for the rest of our predictors. We find that for most of them, by introducing the residual connections, the same effect applies. Increasing number of layers up to two cells for LSTMs and four layers for object-centric transformers has a positive effect, especially for the PSNR metric, and as the prediction horizon increases. The difference in quality between one and four layers of an object-centric transformers increases significantly as the model tries to predict more frames. Evaluation results can be seen in table 5.6.

5.6. Comparisons of Predictors

The higher SAVi performace on Obj3D and the dataset's relative simplicity enable us to perform extensive experimentation on it. When ccomparing the best

Table 5.5: Ablation Study of Residual Vanilla Transformer on Obj3D

	Num Preds = 5			**Num Preds = 15**			**Num Preds = 30**		
	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
Trans. w/ TF	34.27	**0.949**	**0.019**	32.62	0.923	**0.025**	26.67	0.866	0.057
Trans. w/o TF	**34.29**	**0.949**	**0.019**	**32.84**	**0.929**	**0.025**	**29.98**	**0.873**	**0.053**
Trans. w Skip	**34.52**	**0.951**	**0.019**	**32.97**	**0.929**	**0.025**	**30.02**	**0.873**	**0.053**
Trans. w/o Skip	**34.29**	0.949	**0.019**	32.84	**0.929**	**0.025**	29.98	**0.873**	**0.053**
1-Layer w/ Res	**34.23**	**0.949**	**0.020**	**32.62**	**0.926**	**0.026**	**29.79**	**0.869**	**0.055**
1-Layer w/o Res	32.179	0.930	0.025	30.843	0.901	0.034	28.504	0.847	0.071
2-Layers w/ Res	**34.29**	**0.949**	**0.019**	**32.84**	**0.929**	**0.025**	**29.98**	**0.873**	**0.053**
2-Layers w/o Res	32.422	0.933	0.023	31.132	0.906	0.032	28.747	0.852	0.066
4-Layers w/ Res	**34.28**	**0.950**	**0.019**	**32.89**	**0.931**	**0.025**	**30.12**	**0.876**	**0.051**
4-Layers w/ Res	31.914	0.925	0.025	30.385	0.894	0.036	28.287	0.842	0.069
6-Layers w/ Res	**34.17**	**0.949**	**0.019**	**32.79**	**0.929**	**0.025**	**29.93**	**0.873**	**0.053**
6-Layers w/o Res	30.904	0.909	0.028	29.508	0.878	0.04	27.611	0.829	0.078

Table 5.6: Ablation Studies on Recurrent and Object-Centric Modules

	Num Preds = 5			**Num Preds = 15**			**Num Preds = 30**		
	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
LSTM 1-Layer	33.758	0.944	0.021	30.652	0.894	0.04	27.993	**0.836**	0.093
LSTM 2-Layers	**34.209**	**0.948**	**0.02**	**31.125**	**0.9**	**0.039**	**28.131**	0.834	**0.09**
Seq. OCT 1-Layer	34.41	0.950	0.020	32.67	0.925	0.027	29.76	0.867	0.057
Seq. OCT 2-Layers	34.53	**0.951**	**0.019**	33.04	0.931	**0.025**	30.05	0.873	**0.053**
Seq. OCT 4-Layers	**34.55**	**0.951**	**0.019**	**33.10**	**0.932**	**0.025**	**30.15**	**0.875**	**0.053**
Par. OCT 1-Layer	34.07	0.947	0.020	32.43	0.923	0.027	29.60	0.866	0.577
Par. OCT 2-Layers	34.30	**0.950**	**0.019**	32.72	0.928	**0.025**	29.86	0.872	0.055
Par. OCT 4-Layers	**34.31**	0.949	0.020	**32.99**	**0.931**	**0.025**	**30.03**	**0.873**	**0.053**

configurations of all four predictors with each other on Obj3D, it is clear that the transformer predictors in general outperforms the LSTM predictor, especially in the longer prediction horizons. The LSTM can compete with the others in predicting 5 frames, but the difference is too large for the 15 and 30 frame predictions. The comparison of numbers can be seen in table 5.7. We see a visual example of the four predictors on the scene in figure 5.7a. Overall, they all seem to behave reasonably well. Further object-centric video prediction results can be found in Appendix A.

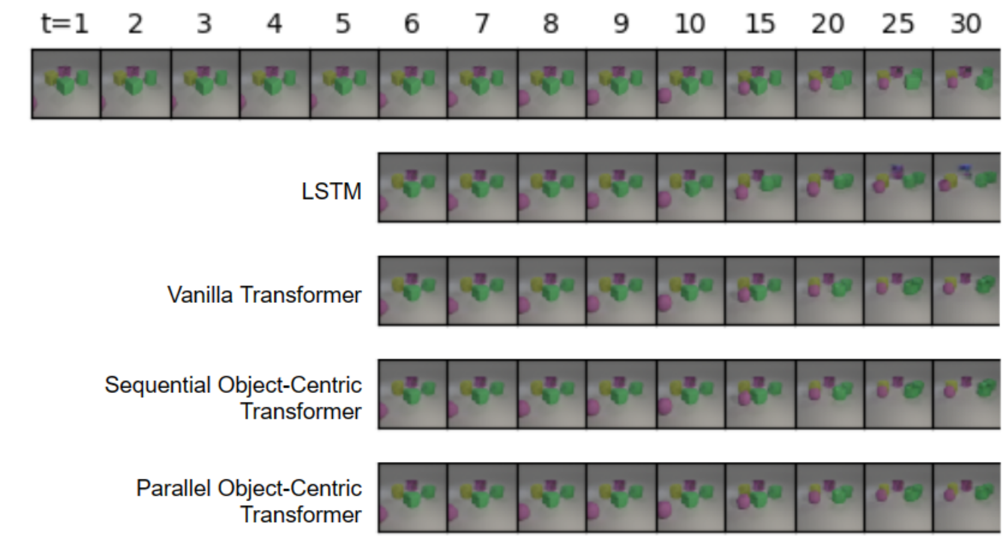
Table 5.7: Quantitative Comparison of Best Predictors on Obj3D.

	Num Preds = 5			**Num Preds = 15**			**Num Preds = 30**		
	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
Vanilla Transformer	34.28	0.950	**0.019**	32.89	0.931	**0.025**	30.12	**0.876**	**0.051**
Seq. OCT	**34.55**	**0.951**	**0.019**	**33.10**	**0.932**	**0.025**	**30.15**	0.875	0.053
Par. OCT	34.31	0.949	0.020	32.99	0.931	**0.025**	30.03	0.873	0.053
LSTM	34.209	0.948	0.02	31.125	0.9	0.039	28.131	0.834	0.09

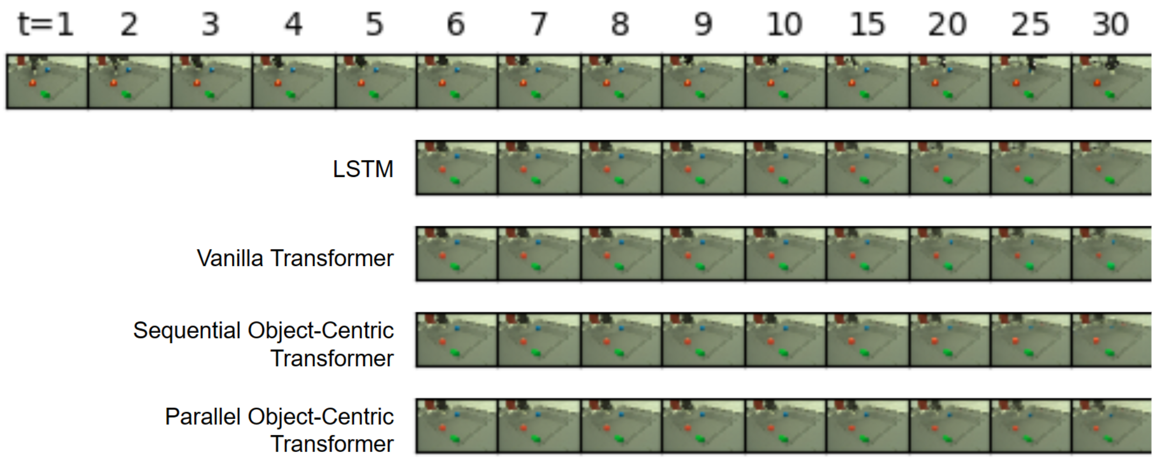
We notice in the animated predictions that the physical boundary of the objects sometimes does not match the visual boundary, leading the objects to happen before the objects touch. It might seem like the objects are acting as repelling magnets. This effect seems to be more prevalent in the LSTM predictions. A reason for this could be the inability of the LSTM predictor we use to model interactions, and as such, while it still handles collisions and occlusions, it does not do it as well as the transformers.

Among the transformers, their performances are very similar, with the sequential object-centric transformer having a slight edge over the others. Still, the vanilla transformer also comes first in some of the metrics, and the parallel object-centric transformer is never far behind. Overall, performance-wise they all perform nearly equally, so it is worth it consider their performances in the light of their complexities and interpretability.

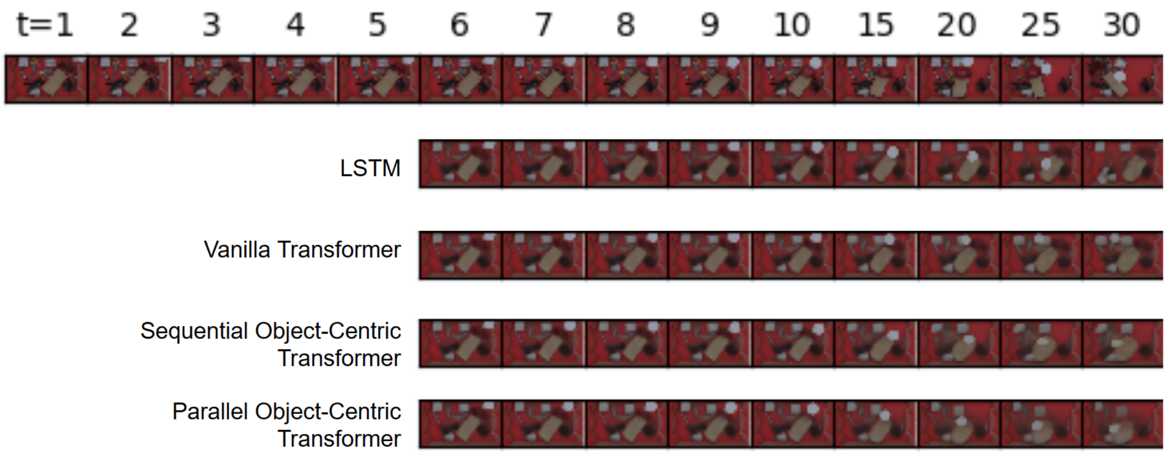
As discussed in the methodology, object-centric transformers are more efficient than a vanilla approach. They enable a batch-wise implementation that divides both the object and time attentions and adds the results, reducing the complexity to a linear one in terms of number of slots and number of frames. In this operation, it only loses the ability to see histories of other slots, while predicting the output for a certain slot. Whether this is a big loss seems to be doubtful, as the performances are quite similar, with the sequential object-centric transformer even performing better sometimes. Visually, we see evidence of this in the attention of the of vanilla transformer as shown in figure 5.10a, where the model mainly attends to other slots


t=1
2
3
4
5
6
7
8
9
10
15
20
25
30
LSTM
Vanilla Transformer
Sequential Object-Centric Transformer
Parallel Object-Centric Transformer

(a) Predictor results on Obj3D.


t=1
2
3
4
5
6
7
8
9
10
15
20
25
30
LSTM
Vanilla Transformer
Sequential Object-Centric Transformer
Parallel Object-Centric Transformer

(b) Predictor results on Sketchy.


t=1
2
3
4
5
6
7
8
9
10
15
20
25
30
LSTM
Vanilla Transformer
Sequential Object-Centric Transformer
Parallel Object-Centric Transformer

(c) Predictor results on SynPick.

Figure 5.7: Example of predictions for the same scene generated by the different predictors.

only in the last time step (4 out of the top 10 attended features), and most of the rest of the attended features are of the slot itself in other time steps (also 4 out of the top 10 attended features). This hints that the introduced object-centric transformers provide the same quality for less complexity. More insights will be provided in the next section on attention masks.

We perform more experiments on the Sketchy and SynPick datasets using the same high performing configurations of the predictors. Qualitative results are shown in figures 5.7b and 5.7c. Metric comparisons are shown in tables 5.8 and 5.9. Note that for Sketchy we extend the prediction horizon to 45 frames, since in this dataset the changes between frames are less prominent and to see change, we needed a higher number of frames.

For Sketchy, the transformers also outperform the LSTM, especially for longer predictions. Here, object-centric transformers perform slightly better than the vanilla transformer, with the parallel object-centric transformer achieving the best results. However, the results are less clear than with Obj3D. For SynPick, this is more of the case, as LSTMs actually achieve the better results in most metrics. It is clear that the lower performance of SAVi affected the training of the predictors. We hypothesize that due to the large number of objects and the complexity of the dataset, slot representations are very imperfect and transformers struggle to learn the interactions from them, which is not the case for LSTMs, which do not explicitly model the interactions.

Table 5.8: Quantitative Comparison on Sketchy.

	Num Preds = 15			**Num Preds = 30**			**Num Preds = 45**		
	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
Vanilla Transformer	23.21	0.860	0.095	22.02	0.835	0.118	21.48	0.817	0.138
Seq. OCT	23.35	**0.863**	0.092	**22.18**	0.837	0.112	21.53	0.813	0.129
Par. OCT	23.288	0.862	**0.0916**	22.14	**0.841**	**0.111**	**21.55**	**0.823**	**0.127**
LSTM	**23.36**	**0.863**	0.094	22.14	0.830	0.118	21.48	0.796	0.146

5.7. Attention Masks

We stated from the beginning that interpretability is one of this thesis' goals, and the attention mechanism are supposed to help this by providing insight into what the model focused on when it was trying to compute a certain output. In this section, we try to qualitatively analyse the attention maps of our transformers in order to extract these insights.

Table 5.9: Quantitative Comparison on Synpick.

	Num Preds = 5			**Num Preds = 15**			**Num Preds = 30**		
	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
Vanilla Transformer	24.08	0.686	0.208	**22.12**	0.629	0.234	20.91	0.572	0.264
Seq. OCT	24.13	**0.687**	0.208	22.09	0.630	0.235	20.81	0.572	0.267
Par. OCT	24.21	0.686	**0.205**	**22.12**	0.624	**0.229**	20.82	0.559	**0.254**
LSTM	**24.23**	**0.687**	0.208	22.10	**0.631**	0.234	**20.92**	**0.578**	0.267

By design, our transformers have different attention structures. For example, the vanilla transformer has one large attention map that spans all objects in the past, while the object-centric transformers have two split maps for each of the time and object dimensions, and even those are split in the implementation in batches. Here we take a look at examples of such maps and try to draw observations from them.

5.7.1. Vanilla Transformer Attention

In figure 5.8, we see an example of the attention maps generated by the vanilla transformer. The effect of the mask that prevents looking into the future, can be clearly seen. Due to the large size of the attention map, it is hard to find any clear observations. However, we can see that whatever object exists in the third slot tends to have the highest attention values.

We check the decomposition of the scene to see what object is at which slot. We see this decomposition in figure 5.9. The object at slot 2 is the biggest object and the one at the center of the scene, which seems to be the reason why the model attends to it frequently. This observation is repeated in other cases.

We go on to check what objects are attended for the most when predicting the state of a certain object. In figure 5.10, we see the complex cases that exist. We have predictions for moving object, stationary object, and the background. The object does not necessarily just look back to the last time step, or its previous states. Multiple factors seem to interact. For example the moving object in figure 5.10c seems to focus more on the nearby object it is about to collide with. The background in figure 5.10b seems to focus on itself more, as its state is naturally not changing. The stationary object in 5.10a is checking its surrounding, that are also mostly static. For static objects that model does not really differentiate between them in the different time steps, because the slots contain nearly the same information.

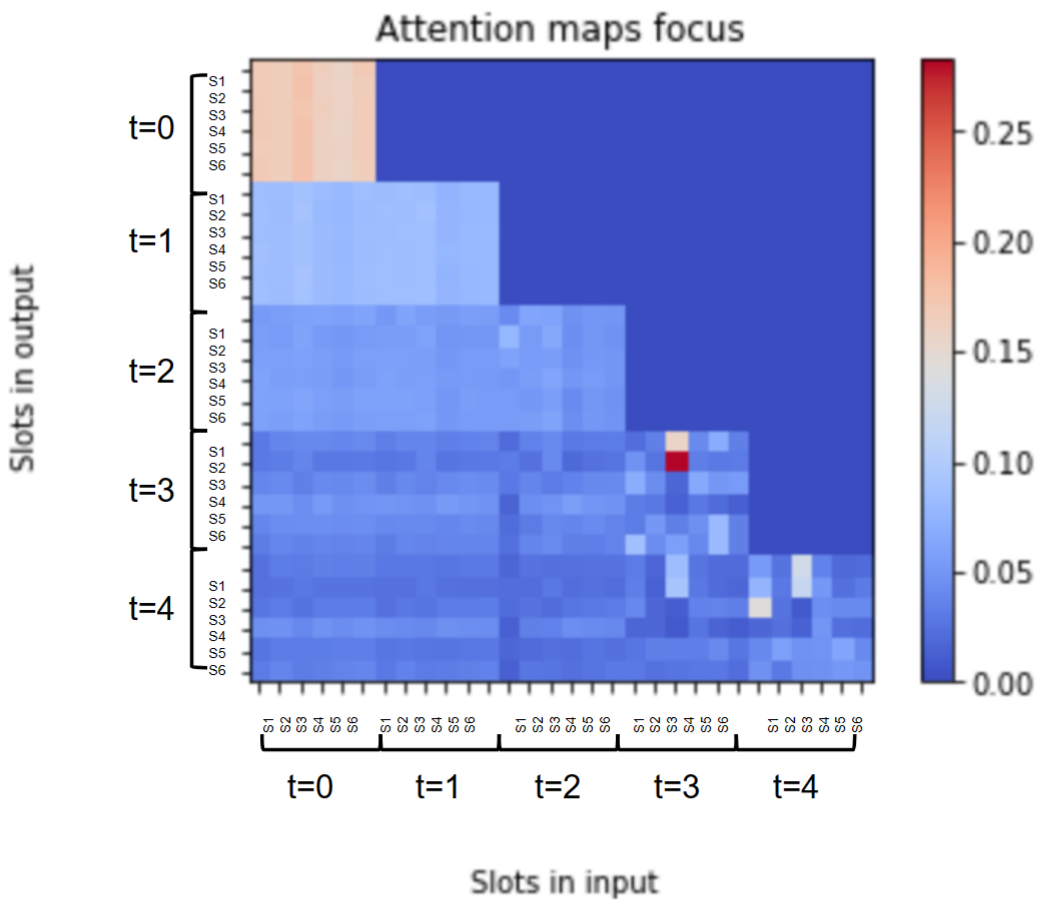


Figure 5.8: Example of a resulting attention map from the vanilla transformer. The input is a 5 time steps seed, with each time stop containing 6 slots, building an input of 30 slots in total.

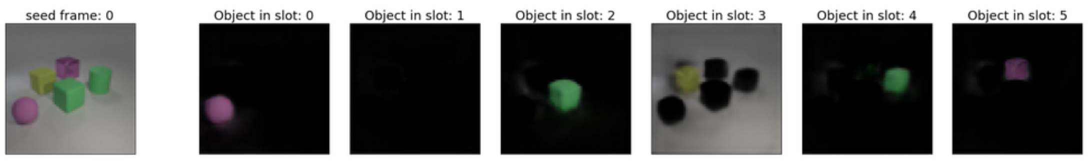
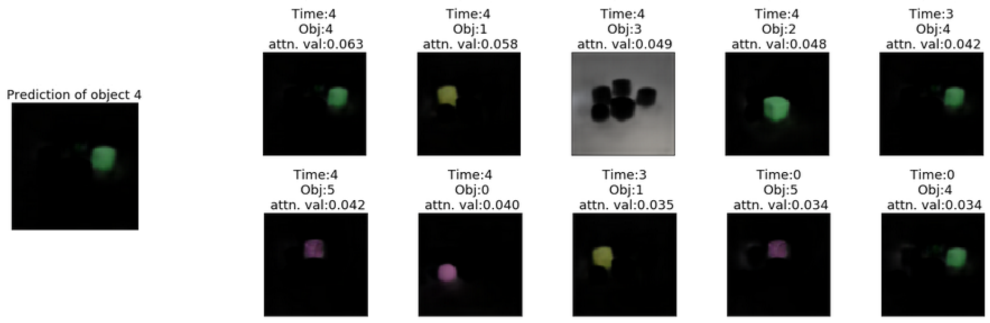
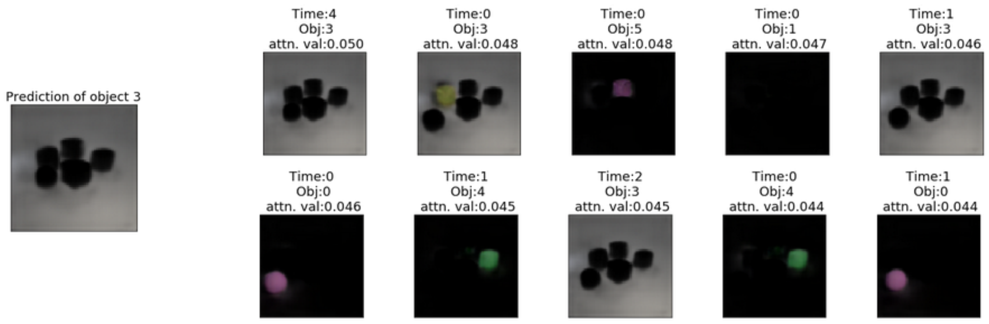


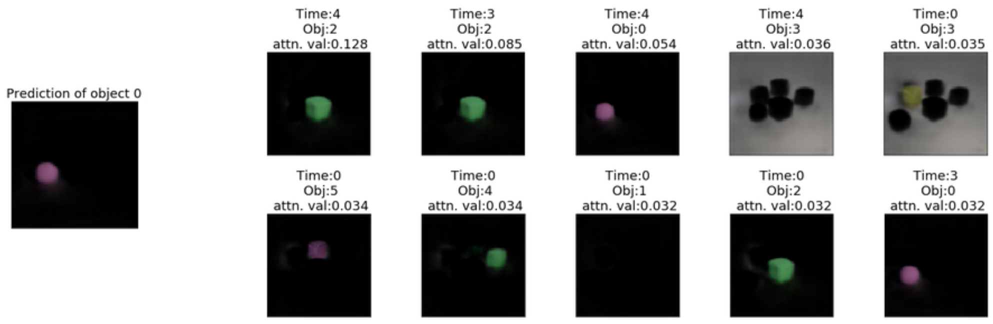
Figure 5.9: Object decomposition in a scene, where each object was assigned to a slot. We notice that slot remained empty while an object that should have been in it is grouped with the background. A reason for this could be that the object is in the back and of near colour to the background.


Prediction of object 4
Time:4 Obj:4 attn. val:0.063
Time:4 Obj:1 attn. val:0.058
Time:4 Obj:3 attn. val:0.049
Time:4 Obj:2 attn. val:0.048
Time:3 Obj:4 attn. val:0.042
Time:4 Obj:5 attn. val:0.042
Time:4 Obj:0 attn. val:0.040
Time:3 Obj:1 attn. val:0.035
Time:0 Obj:5 attn. val:0.034
Time:0 Obj:4 attn. val:0.034

(a) Vanilla transformer attention for a stationary object.


Prediction of object 3
Time:4 Obj:3 attn. val:0.050
Time:0 Obj:3 attn. val:0.048
Time:0 Obj:5 attn. val:0.048
Time:0 Obj:1 attn. val:0.047
Time:1 Obj:3 attn. val:0.046
Time:0 Obj:0 attn. val:0.046
Time:1 Obj:4 attn. val:0.045
Time:2 Obj:3 attn. val:0.045
Time:0 Obj:4 attn. val:0.044
Time:1 Obj:0 attn. val:0.044

(b) Vanilla transformer attention for the background.


Prediction of object 0
Time:4 Obj:2 attn. val:0.128
Time:3 Obj:2 attn. val:0.085
Time:4 Obj:0 attn. val:0.054
Time:4 Obj:3 attn. val:0.036
Time:0 Obj:3 attn. val:0.035
Time:0 Obj:5 attn. val:0.034
Time:0 Obj:4 attn. val:0.034
Time:0 Obj:1 attn. val:0.032
Time:0 Obj:2 attn. val:0.032
Time:3 Obj:0 attn. val:0.032

(c) Vanilla transformer attention for a moving object.

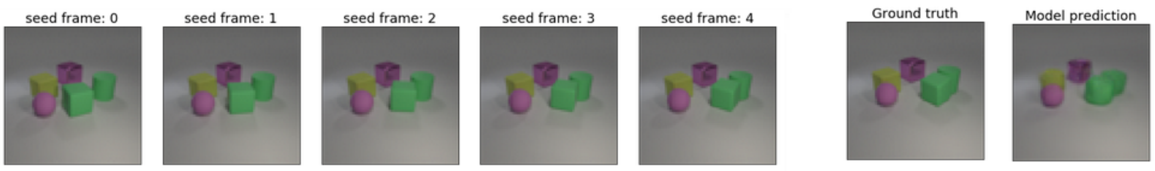
Figure 5.10: Example of slot attention of vanilla transformer while predicting certain slots. We show the predicted slots, top 10 slots used in prediction, with their corresponding slot number, time step, and attention value.

5.7.2. Object-Centric Transformers Attention

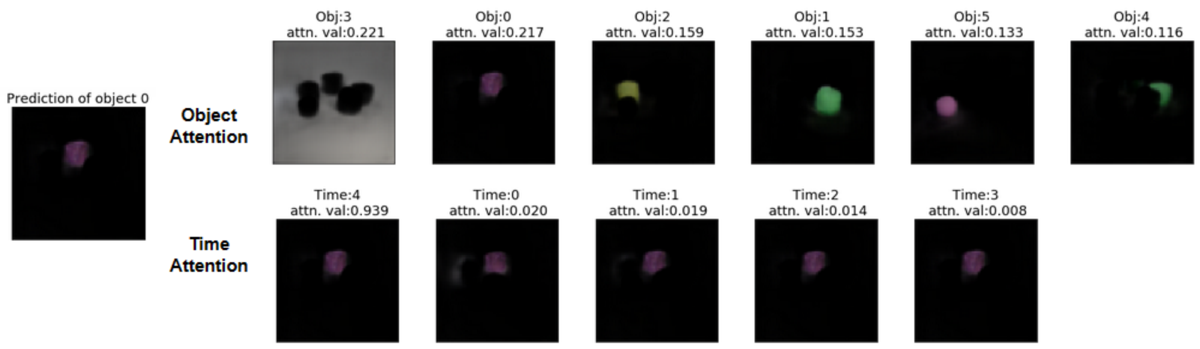
In order to take a look at object-centric attentions, we take a look at the prediction as a whole, at attention values for individual objects for both object and time attentions, and the corresponding attention masks. In figures 5.11 and 5.12 we see detailed examples of the attention computation in both these predictors. We also observe some of their working mechanisms, and notice how similarly they both behave.

For time attention, both models focus more on the last and the first time steps. The model draws the prediction from the initial and final states, which might be because states in between are included in their difference. The attention at the last time step is the highest by a large margin, which is logical since this is the state the object is coming from. Differences between attention values for the rest of the time steps are much less significant. The order of time steps in time attention tends to be equal between the two predictors.

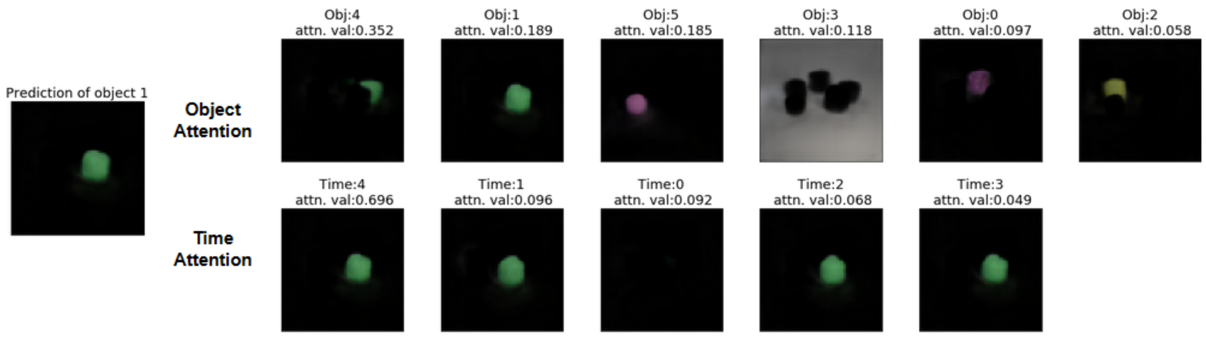
For object attention, the attention values are distributed more over the slots. It does not show a clear criteria for how the models tend to choose objects to attend to, but it seems the distance between objects play a role in this, which is to be expected. The nearer an object is, the more it is likely to affect the current object. Object also do not seem to always attend to themselves during prediction, which should be due to the existence of time attention. If the object has seen all its versions in the past, the information offered from it in the object attention is just a repetition.



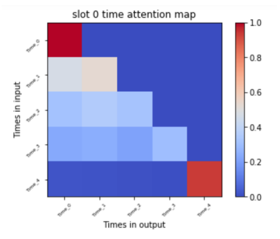
(a) Seed frames, the GT and the prediction.



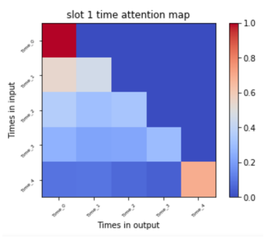
(b) Example of attention values for predicting object 0.



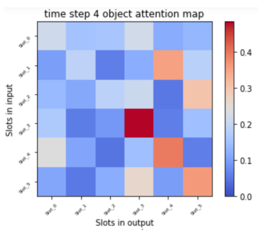
(c) Example of attention values for predicting object 1.



(d) Time attention of object in slot 0.

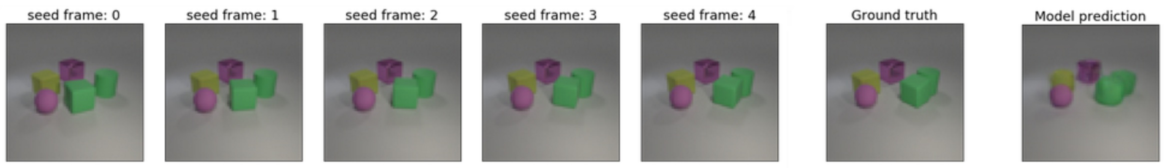


(e) Time attention of object in slot 1.

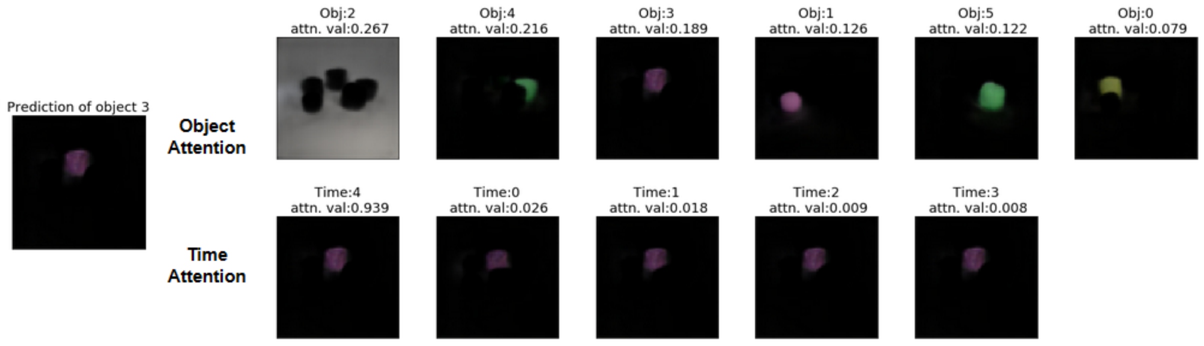


(f) Object attention map at last time step.

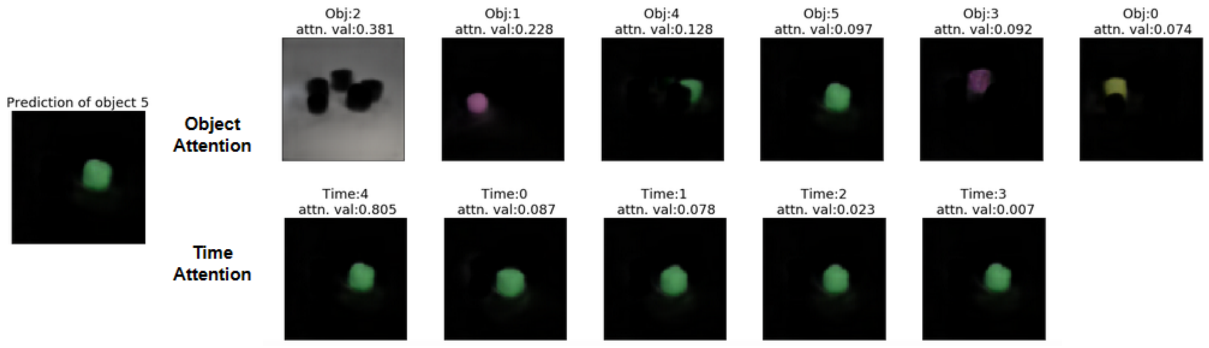
Figure 5.11: Summary of the attention process of the sequential object-centric transformer.



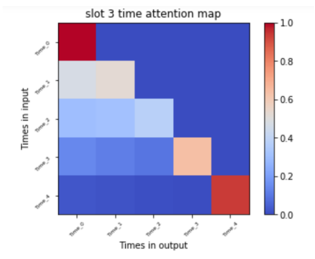
(a) Seed frames, the GT and the prediction.



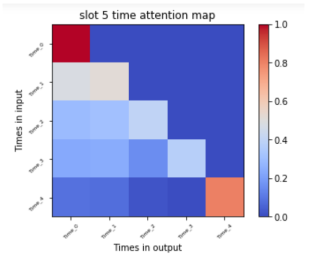
(b) Example of attention values for predicting same object of figure 5.11b.



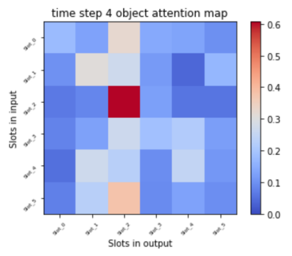
(c) Example of attention values for predicting same object of figure 5.11c.



(d) Time attention of slot 3 (same object in figure 5.11d).



(e) Time attention of slot 5 (same object in figure 5.11e).



(f) Object attention map at last time step.

Figure 5.12: Summary of the attention process of the parallel object-centric transformer.

6. Conclusion

The goal of this thesis was to fuse object-centric learning with video prediction, to provide interpretable and explainable results. We set out to look for an efficient method for extracting information into objects, applying prediction onto said objects, and reconstruct back the frames. We investigated the combination of such object-centric representations with different predictor models, and investigated how they work together.

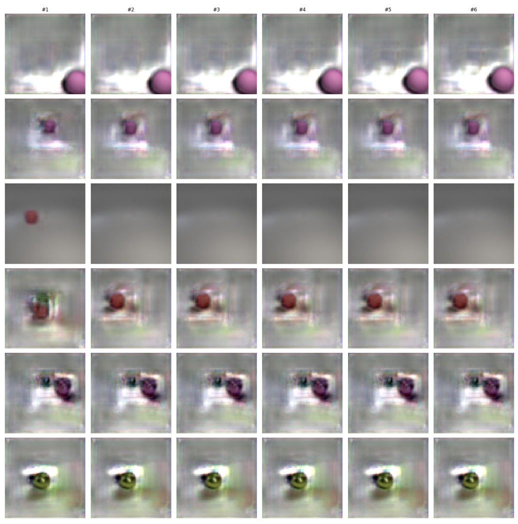
Our starting point was the slot attention mechanism, an approach applicable for both images and videos, to learn about object in a scene, and encoding them separately, while having the possibility to recombine these representations again. Using Slot Attention For Video (SAVi), we trained multiple models on multiple datasets, achieving different degrees of success. Using these models we trained four types of predictors, exploring attention for modeling object dynamics and interactions in four different ways. The LSTM model has no attention and had no way of modelling the interaction between slots, as it could only look at the slots' own history. The vanilla transformer had a complex attention span, looking at the complete history of all slots together, and allowed to connect a certain slot at a certain time step with another one at another time step. The object-centric transformers were allowed to look in rows and columns, i.e., for each slot its own history, or the current state of the others. The first version did this sequentially, allowing the time attention part to take a look at supplemented input from the object attention part, whereas the second version did this in parallel, separating the two flows completely.

Our goal of having a fully self-superivsed approach is achieved. For two out of the three datasets, namely Obj3D and Sketchy, complete self-supervision is granted. The data we used did not have labelling in any way in both phases of training. Our predictors were able to achieve good results on the Obj3D dataset. They show that given good object representations, they are able to predict frames of good quality and sensible scenarios. We show that the attention between slots lead to better results, with the gap increasing with the number of predicted frames, and that this attention does not have to have high complexity. Just focusing on the object-centric version can lead to results of the same quality, with less computational needs. We show that the attention models can be used for analysing and visualizing the prediction process. Our object-centric transformers provide easier to understand and visualize attention maps, because they span a smaller space and can be interpreted in terms of two questions ”what parts of my history I see as important for my future?” and ”what part of my surrounding I think will affect me in the future?”. Combing both provides valuable insights into the models' workings.

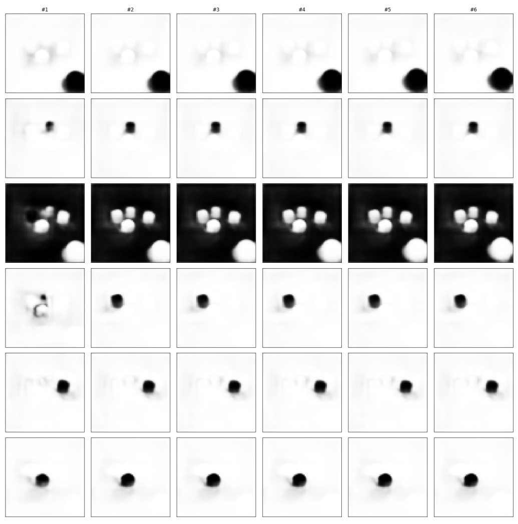
Still, we also see there is plenty of room for improvement in our approach. The predictors are dependant on having a well trained SAVi results with good results, which needs some experimentation to configure all the possible settings, such as number of slot attention iterations, number of slots, dimensionality of slots, and capacity of encoders, each of them coming with their computational costs. For more complicated datasets, such as Sketchy and SynPick, we need to more time and resources to test them and find a good enough combination of settings that can handle their complexities, which means weeks more of training that we did not have for this thesis. Moreover, we sometimes see artifacts in our predictions, such as force fields that prevent objects from touching as they collide. These need more investigation regarding their causes and solutions. Our model also does not seem to handle highly uncertain scenarios, such is the case for robotics datasets, where the motion is governed by intention of the operator, not the dynamics.

There is a lot of exciting suggestions for future works. First of all, more experimentation with slot initialization and even introducing some object information could be helpful for extracting objects especially in the harder datasets. This was already needed in SynPick and it could be worth it to experiment with easy to obtain labelling data. In this regard, we see that complex datasets can lead SAVi to attend to frame patches instead of objects, so having object information in slot initialization could help there. Moreover, we could extend the training to be fully end-to-end, so that the predictor is trained directly with SAVi, optimizing the representations directly for prediction. It also might be worth it to experiment with using different loss functions or the same functions but with different weights for each component. Overall, experimenting more and trying to achieve good results on more datasets will help us understand how the model works and how it can be improved. For datasets with intentional movements, introducing mutli-modality in our predictions using a stochastic component could help the model understand how the future is shaped and can change. Another way to handle this could be collecting action input, e.g. input given to gripper that leads to its movement in robotics datasets, and then giving this input to the model, which would be another interesting experiment.

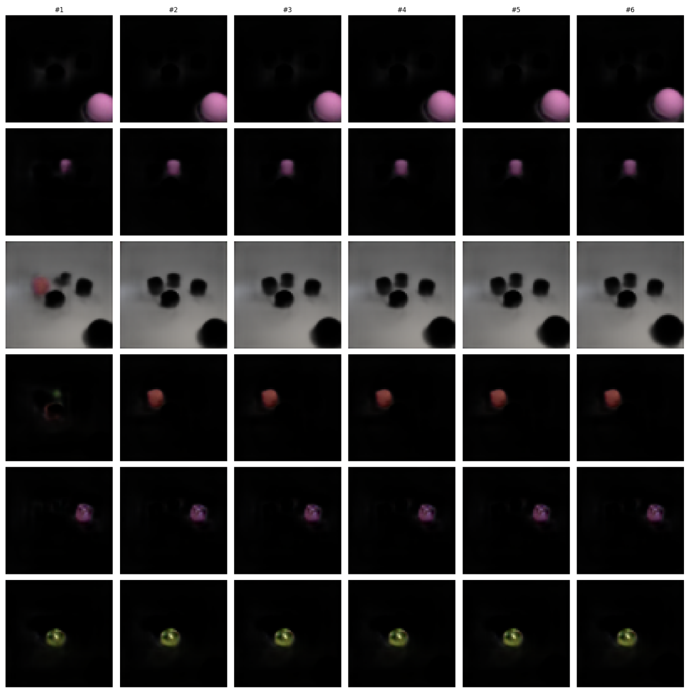
A. Appendix



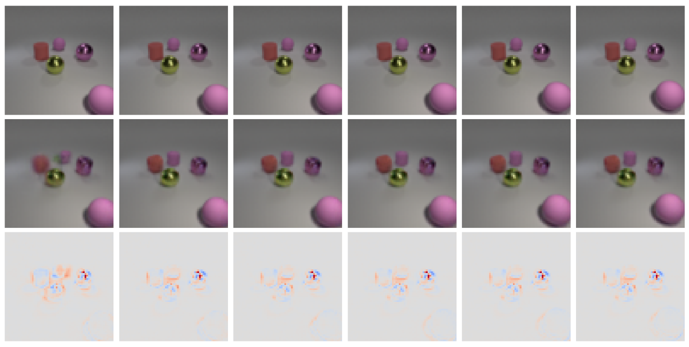
(a) Decoded slots into objects.



(b) Masks generated for decoding.

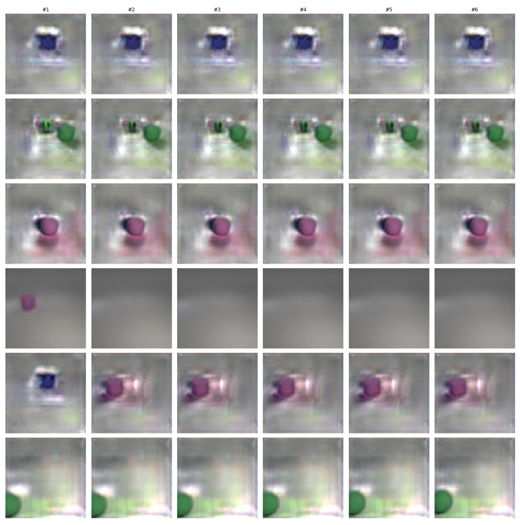


(c) Combination of decoded objects and masks.

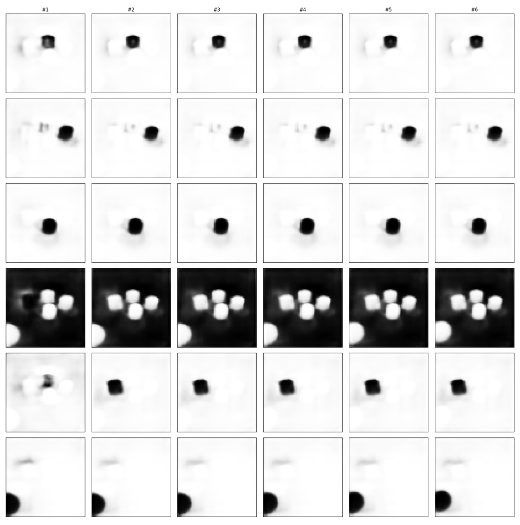


(d) Combined scene reconstruction.

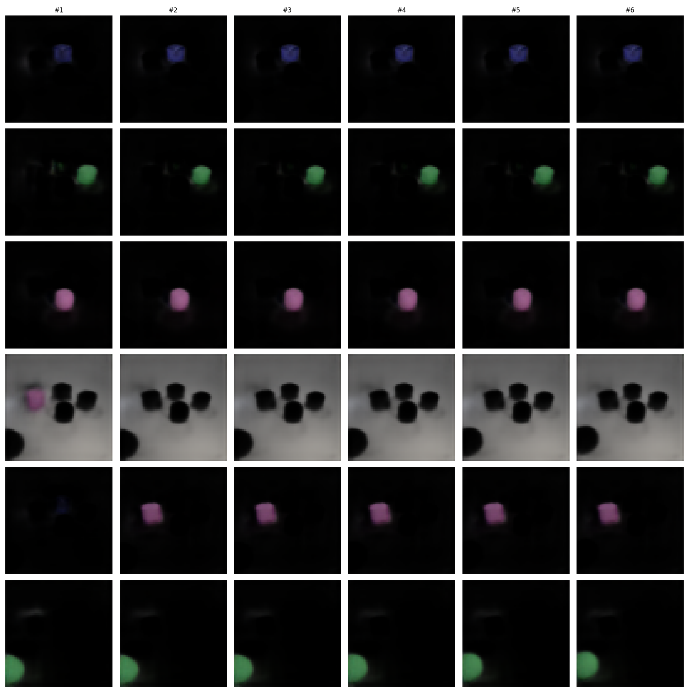
Figure A.1: Example 1 of decoding flow from slots to frame on Obj3D.



(a) Decoded slots into objects.



(b) Masks generated for decoding.



(c) Combination of decoded objects and masks.

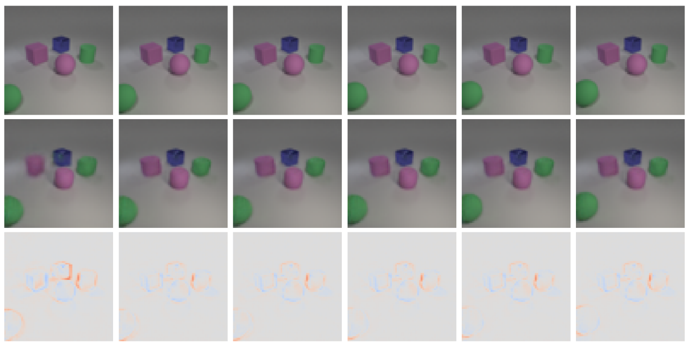
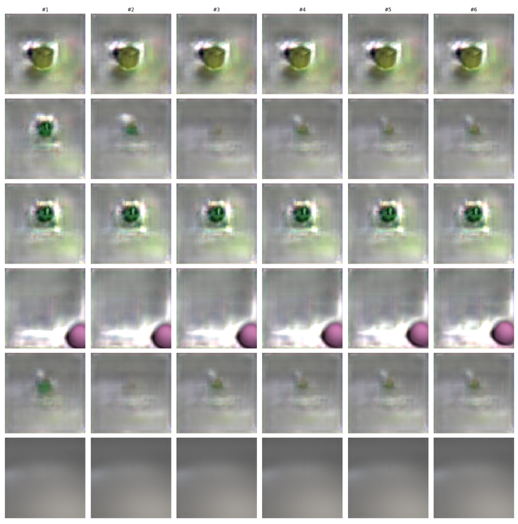
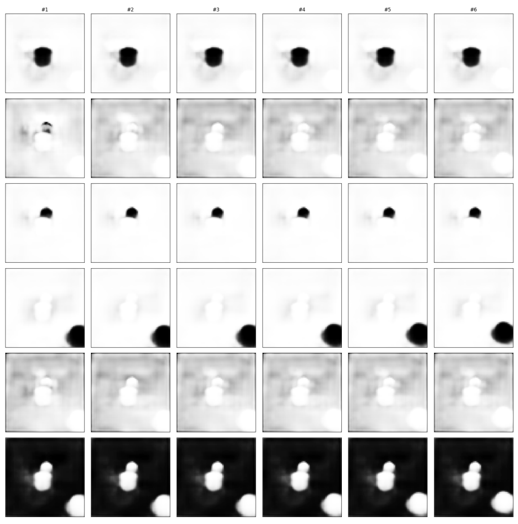

(d) Combined scene reconstruction.

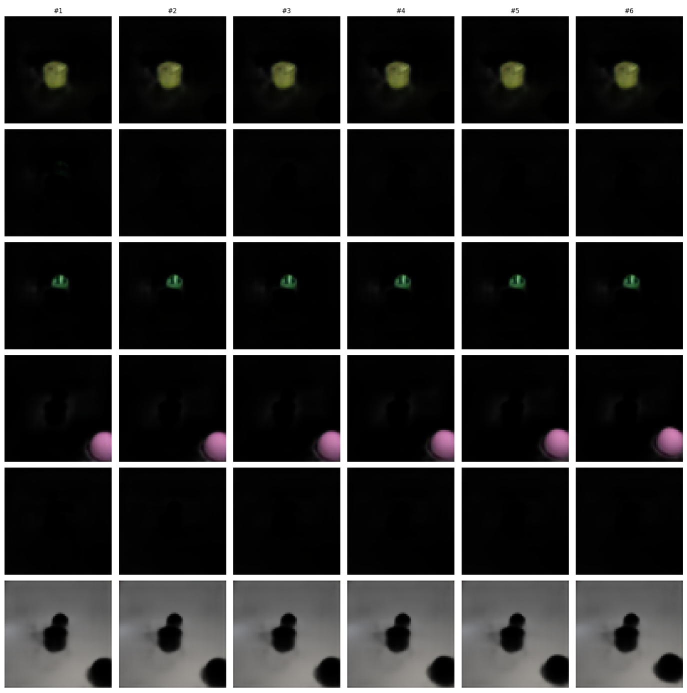
Figure A.2: Example 2 of decoding flow from slots to frame on Obj3D.



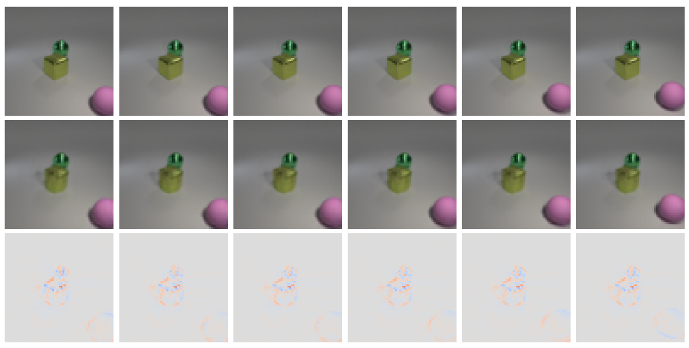
(a) Decoded slots into objects.



(b) Masks generated for decoding.



(c) Combination of decoded objects and masks.



(d) Combined scene reconstruction.

Figure A.3: Example 3 of decoding flow from slots to frame on Obj3D.

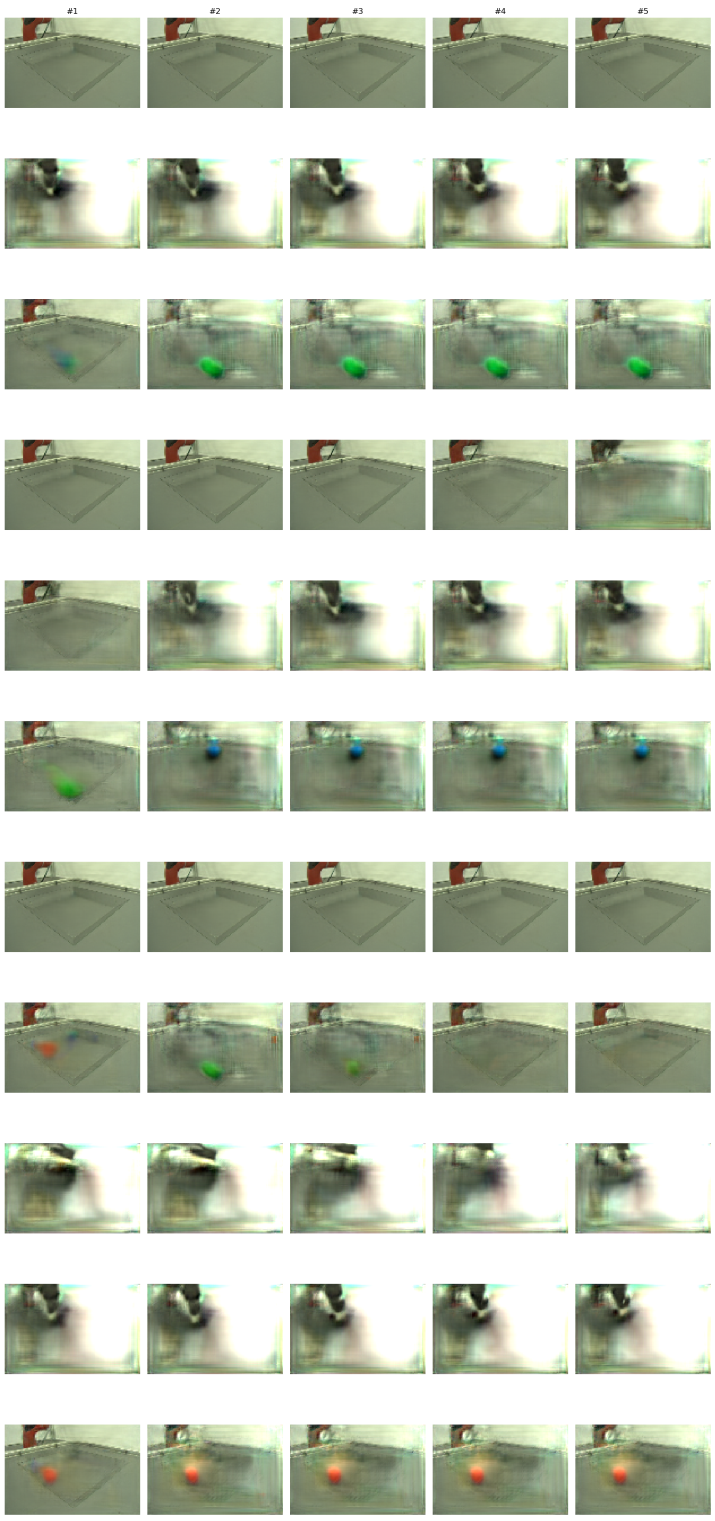


Figure A.4: Example of slot decoding in our trained instance of SAVi on Sketchy dataset. Original reconstructions shown in figure 5.6b.

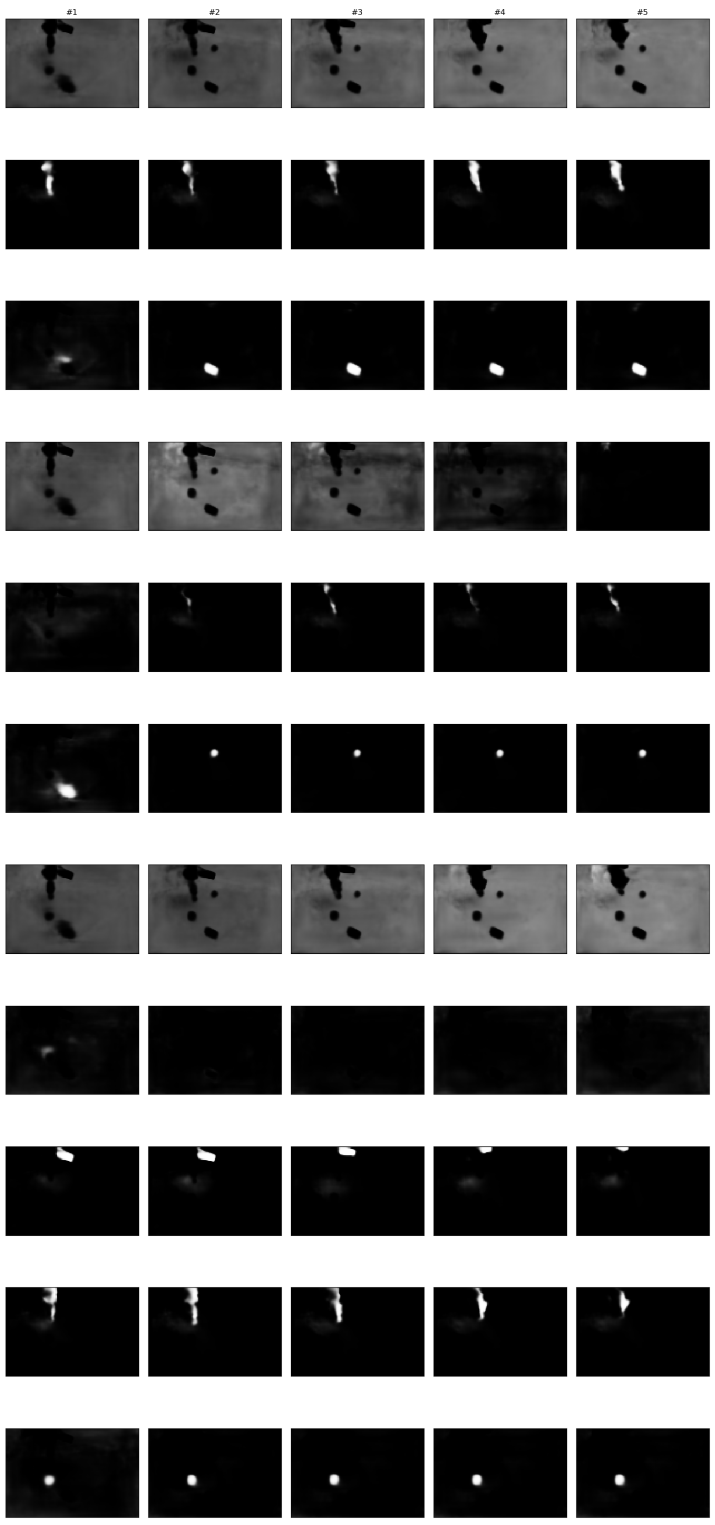


Figure A.5: Example of masks generated in our trained instance of SAVi on Sketchy dataset. Original reconstructions shown in figure 5.6b.

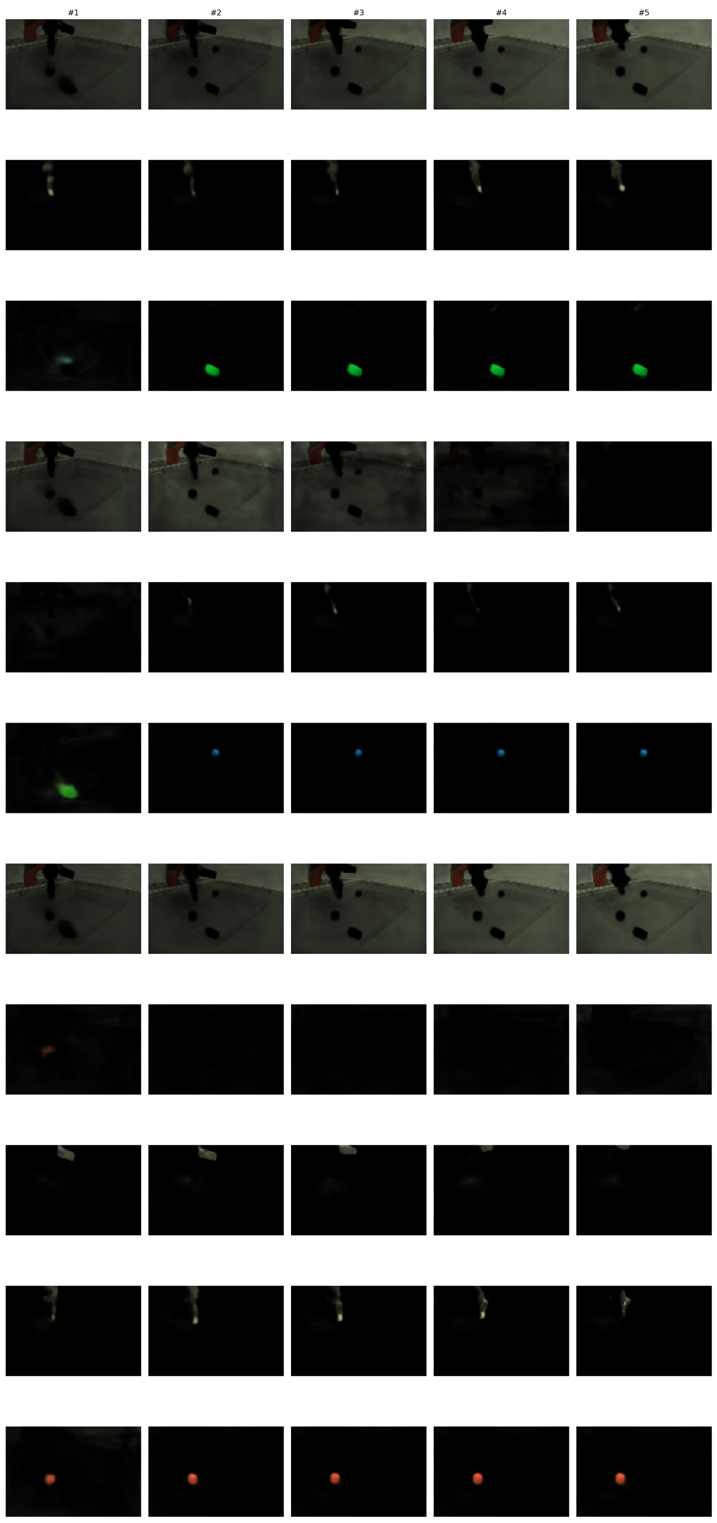
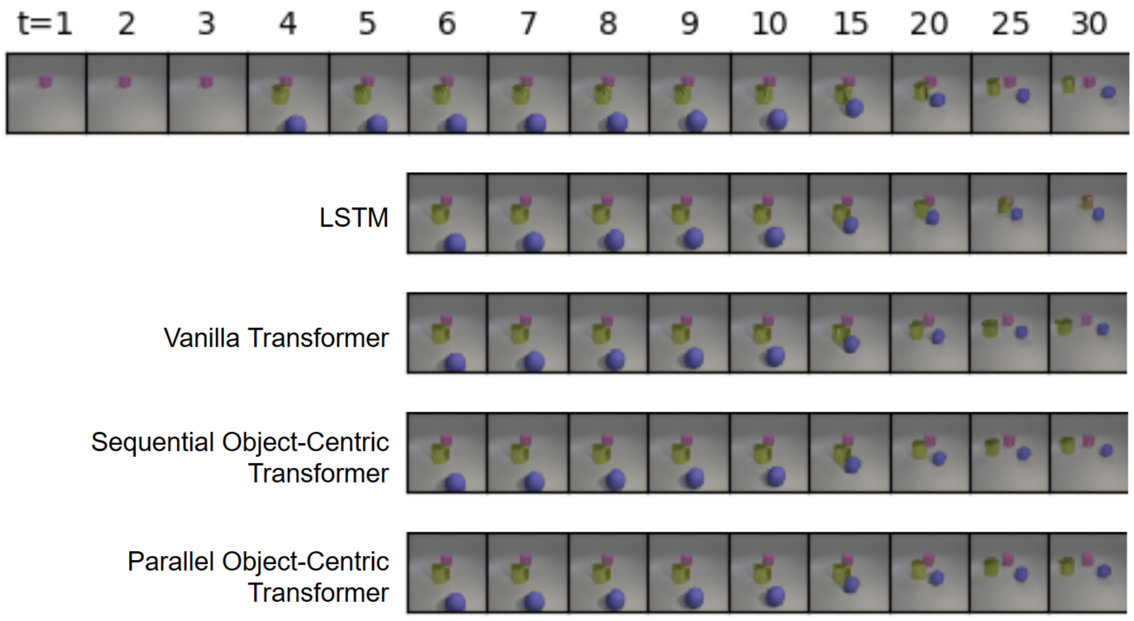
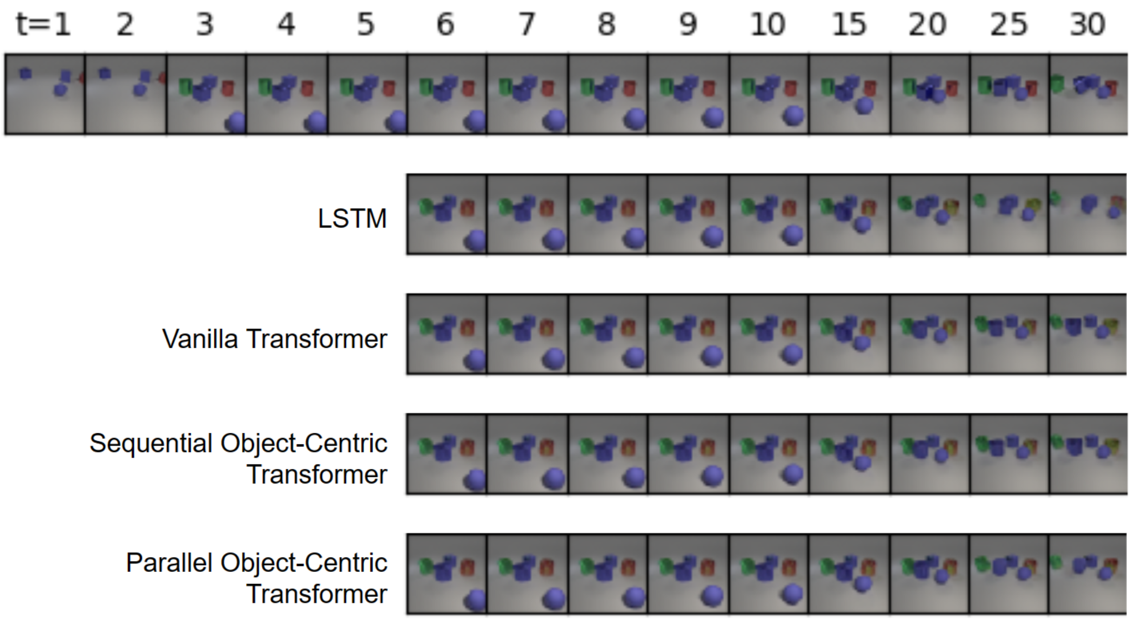


Figure A.6: Example of combining masks with generated objects in our trained instance of SAVi on Sketchy dataset. Original reconstructions shown in figure 5.6b.


t=1
2
3
4
5
6
7
8
9
10
15
20
25
30
LSTM
Vanilla Transformer
Sequential Object-Centric Transformer
Parallel Object-Centric Transformer

(a)


t=1
2
3
4
5
6
7
8
9
10
15
20
25
30
LSTM
Vanilla Transformer
Sequential Object-Centric Transformer
Parallel Object-Centric Transformer

(b)

Figure A.7: Predictor results on Obj3D.

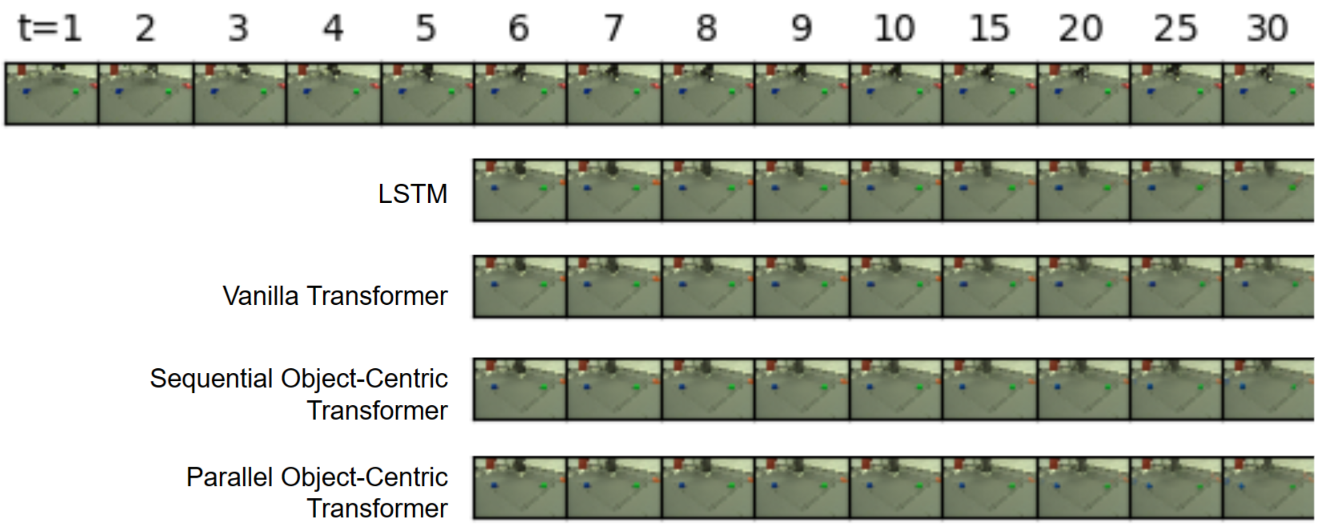

t=1
2
3
4
5
6
7
8
9
10
15
20
25
30
LSTM
Vanilla Transformer
Sequential Object-Centric Transformer
Parallel Object-Centric Transformer

Figure A.8: Predictions on Sketchy.

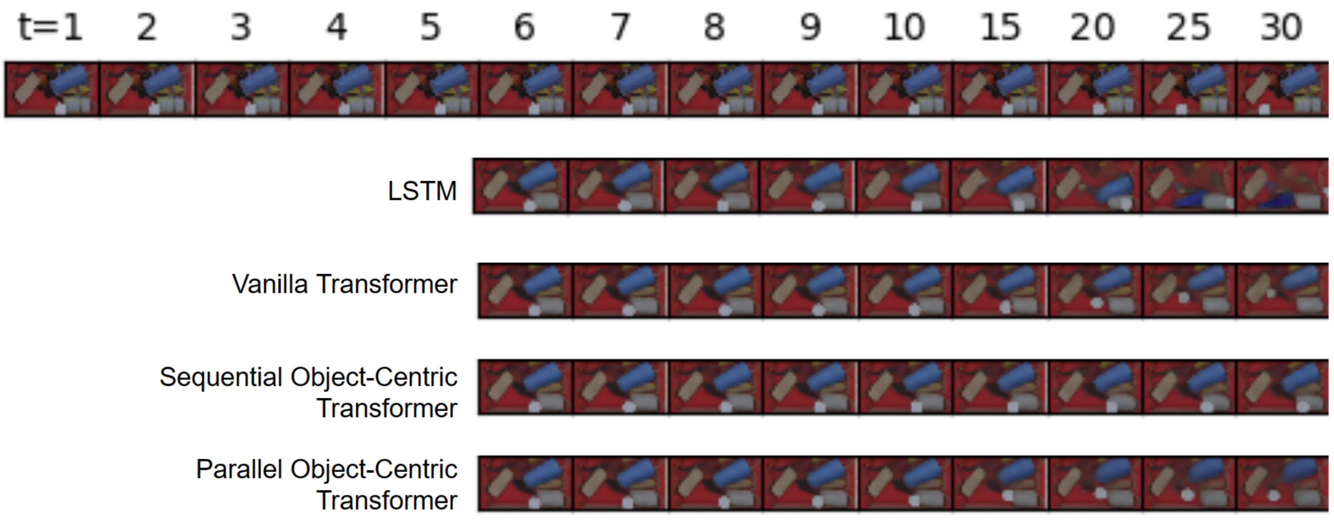

t=1
2
3
4
5
6
7
8
9
10
15
20
25
30
LSTM
Vanilla Transformer
Sequential Object-Centric Transformer
Parallel Object-Centric Transformer

Figure A.9: Predictions on SynPick.

List of Figures

List of Tables

Bibliography

Adadi, Amina and Mohammed Berrada (2018). “Peeking Inside the Black-box: a Survey on Explainable Artificial Intelligence (XAI).” In: *IEEE access*.

Akbik, Alan, Duncan Blythe, and Roland Vollgraf (2018). “Contextual String Embeddings for Sequence Labeling.” In: *Proceedings of the 27th international conference on computational linguistics*.

Badrinarayanan, Vijay, Alex Kendall, and Roberto Cipolla (2017). “Segnet: A Deep Convolutional Encoder-decoder Architecture for Image Segmentation.” In: *IEEE transactions on pattern analysis and machine intelligence*.

Behrmann, Nadine, Jurgen Gall, and Mehdi Noroozi (2021). “Unsupervised Video Representation Learning by Bidirectional Feature Prediction.” In: *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*.

Benaim, Sagie, Ariel Ephrat, Oran Lang, Inbar Mosseri, William T Freeman, Michael Rubinstein, Michal Irani, and Tali Dekel (2020). “Speednet: Learning the Speediness in Videos.” In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

Burgess, Christopher P, Loic Matthey, Nicholas Watters, Rishabh Kabra, Irina Higgins, Matt Botvinick, and Alexander Lerchner (2019a). “MoNet: Unsupervised scene decomposition and representation.” In: *ArXiv preprint arXiv:1901.11390*.

– (2019b). “Monet: Unsupervised Scene Decomposition and Representation.” In: *ArXiv preprint arXiv:1901.11390*.

Cabi, Serkan, Sergio Gómez Colmenarejo, Alexander Novikov, Ksenia Konyushkova, Scott Reed, Rae Jeong, Konrad Zolna, Yusuf Aytar, David Budden, Mel Vecerik, Oleg Sushkov, David Barker, Jonathan Scholz, Misha Denil, Nando de Freitas, and Ziyu Wang (2019). “Scaling data-driven robotics with reward sketching and batch reinforcement learning.” In: *Robotics: Science and Systems (RSS)*.

Char, Danton S, Nigam H Shah, and David Magnus (2018). “Implementing Machine Learning in Health Care — Addressing Ethical Challenges.” In: *The New England journal of medicine*.

Chen, Mia Xu, Orhan Firat, Ankur Bapna, Melvin Johnson, Wolfgang Macherey, George Foster, Llion Jones, Niki Parmar, Mike Schuster, Zhifeng Chen, et al. (2018). “The Best of Both Worlds: Combining Recent Advances in Neural Machine Translation.” In: *Annual Meeting of the Association for Computational Linguistics (ACL)*.

Chiu, Hsu-kuang, Ehsan Adeli, and Juan Carlos Niebles (2020). “Segmenting the Future.” In: *IEEE Robotics and Automation Letters*.

Cho, Kyunghyun, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio (2014). “On the Properties of Neural Machine Translation: Encoder-decoder Approaches.” In: *Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation (SSST-8).*

Cho, Kyunghyun, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio (2014). “Learning Phrase Representations using RNN Encoder-decoder for Statistical Machine Translation.” In: *Conference on Empirical Methods in Natural Language Processing (EMNLP).*

Deloche, François (2017). *Unfolded basic recurrent neural network.* URL: `https://commons.wikimedia.org/wiki/%7BF%7Dile:%7BR%7Decurrent_neural_network_unfold.svg#/media/%7BF%7Dile:%7BR%7Decurrent_neural_network_unfold.svg`.

Deng, Jia, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei (2009). “Imagenet: A Large-scale Hierarchical Image Database.” In: *2009 IEEE conference on computer vision and pattern recognition.* Ieee.

Denton, Emily and Rob Fergus (2018). “Stochastic video generation with a learned prior.” In: *International Conference on Machine Learning.* PMLR.

Diba, Ali, Vivek Sharma, Luc Van Gool, and Rainer Stiefelhagen (2019). “Dynamonet: Dynamic Action and Motion Network.” In: *Proceedings of the IEEE/CVF International Conference on Computer Vision.*

Dosovitskiy, Alexey, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. (2021). “An image is worth 16x16 words: Transformers for image recognition at scale.” In: *International Conference on Learning Representations (ICLR).*

Farazi, Hafez and Sven Behnke (2020). “Motion Segmentation Using Frequency Domain Transformer Networks.” In: *European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN).*

Farazi, Hafez, Jan Nogga, et al. (2021). “Semantic Prediction: Which One Should Come First, Recognition or Prediction?” In: *European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN).*

Farazi, Hafez, Jan Nogga, and Sven Behnke (2021). “Local Frequency Domain Transformer Networks for Video Prediction.” In: *2021 International Joint Conference on Neural Networks (IJCNN).* IEEE.

Finn, Chelsea, Ian Goodfellow, and Sergey Levine (2016). “Unsupervised Learning for Physical Interaction Through Video Prediction.” In: *Advances in Neural Information Processing Systems.*

Gao, Zhangyang, Cheng Tan, Lirong Wu, and Stan Z Li (2022). “SimVP: Simpler Yet Better Video Prediction.” In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition.*

Goodman, Bryce and Seth Flaxman (2017). “European Union Regulations on Algorithmic Decision-making and a “Right to Explanation”.” In: *AI magazine.*

Green, Edwin James and Jake Quilty-Dunn (2021). "What is an Object File?" In: *The British Journal for the Philosophy of Science.*

Greff, Klaus, Raphaël Lopez Kaufman, Rishabh Kabra, Nick Watters, Christopher Burgess, Daniel Zoran, Loic Matthey, Matthew Botvinick, and Alexander Lerchner (2019). "Multi-object Representation Learning with iterative Variational Inference." In: *International Conference on Machine Learning.* PMLR.

Greff, Klaus, Sjoerd Van Steenkiste, and Jürgen Schmidhuber (2020). "On the Binding Problem in Artificial Neural Networks." In: *ArXiv preprint arXiv:2012.05208.*

Harley, Adam W, Yiming Zuo, Jing Wen, Ayush Mangal, Shubhankar Potdar, Ritwick Chaudhry, and Katerina Fragkiadaki (2021). "Track, Check, Repeat: An EM Approach to Unsupervised Tracking." In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition.*

He, Kaiming, Xiangyu Zhang, Shaoqing Ren, and Jian Sun (2016). "Deep Residual Learning for Image Recognition." In: *Proceedings of the IEEE conference on computer vision and pattern recognition.*

Herdade, Simao, Armin Kappeler, Kofi Boakye, and Joao Soares (2019). "Image Captioning: Transforming Objects Into Words." In: *Advances in Neural Information Processing Systems.*

Hochreiter, Sepp and Jürgen Schmidhuber (1997). "Long short-term memory." In: *Neural computation.*

Hossain, MD Zakir, Ferdous Sohel, Mohd Fairuz Shiratuddin, and Hamid Laga (2019). "A Comprehensive Survey of Deep Learning for Image Captioning." In: *ACM Computing Surveys (CsUR).*

Huang, Gao, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger (2017). "Densely Connected Convolutional Networks." In: *Proceedings of the IEEE conference on computer vision and pattern recognition.*

Jaderberg, Max, Karen Simonyan, Andrew Zisserman, et al. (2015). "Spatial Transformer Networks." In: *Advances in Neural Information Processing Systems.*

Jiang, Jindong, Sepehr Janghorbani, Gerard De Melo, and Sungjin Ahn (2019). "Scalor: Generative World Models with Scalable Object Representations." In: *ArXiv preprint arXiv:1910.02384.*

Jing, Longlong, Xiaodong Yang, Jingen Liu, and Yingli Tian (2018). "Self-supervised Spatiotemporal Feature Learning via Video Rotation Prediction." In: *ArXiv preprint arXiv:1811.11387.*

Johnson, Justin, Bharath Hariharan, Laurens Van Der Maaten, Li Fei-Fei, C Lawrence Zitnick, and Ross Girshick (2017). "Clevr: A diagnostic dataset for compositional language and elementary visual reasoning." In: *Proceedings of the IEEE conference on computer vision and pattern recognition.*

Johnson, Scott P (2018). "Object perception." In: *Oxford Research Encyclopedia of Psychology.*

Kahneman, Daniel, Anne Treisman, and Brian J Gibbs (1992). "The Reviewing of Object Files: Object-specific Integration of Information." In: *Cognitive psychology.*

Karapetyan, Ani, Angel Villar-Corrales, Andreas Boltres, and Sven Behnke (2022). "Video Prediction at Multiple Scales with Hierarchical Recurrent Networks." In: *ArXiv preprint arXiv:2203.09303*.

Kim, Dahun, Donghyeon Cho, and In So Kweon (2019). "Self-supervised Video Representation Learning with Space-time Cubic Puzzles." In: *Proceedings of the AAAI conference on artificial intelligence*.

Kingma, Diederik P and Max Welling (2013). "Auto-encoding Variational Bayes." In: *International Conference on Learning Representations (ICLR)*.

Kipf, Thomas, Gamaleldin F Elsayed, Aravindh Mahendran, Austin Stone, Sara Sabour, Georg Heigold, Rico Jonschkowski, Alexey Dosovitskiy, and Klaus Greff (2022). "Conditional object-centric learning from video." In: *International Conference on Learning Representations (ICLR)*.

Kipf, Thomas, Ethan Fetaya, Kuan-Chieh Wang, Max Welling, and Richard Zemel (2018). "Neural Relational Inference for Interacting Systems." In: *International Conference on Machine Learning*. PMLR.

Kossen, Jannik, Karl Stelzner, Marcel Hussing, Claas Voelcker, and Kristian Kersting (2019). "Structured Object-aware Physics Prediction for Video Modeling and Planning." In: *International Conference on Learning Representations (ICLR)*.

Kuhn, Harold W. (1955). "The Hungarian Method for the Assignment Problem." In: *Naval Research Logistics Quarterly* 2.

Lin, Tsung-Yi, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick (2014). "Microsoft COCO: Common Objects in Context." In: *European conference on computer vision*. Springer.

Lin, Zhixuan, Yi-Fu Wu, Skand Peri, Bofeng Fu, Jindong Jiang, and Sungjin Ahn (2020a). "Improving Generative Imagination in Object-centric World Models." In: *International Conference on Machine Learning*. PMLR.

– (2020b). "Improving generative imagination in object-centric world models." In: *International Conference on Machine Learning*. PMLR.

Lin, Zhixuan, Yi-Fu Wu, Skand Vishwanath Peri, Weihao Sun, Gautam Singh, Fei Deng, Jindong Jiang, and Sungjin Ahn (2020). "Space: Unsupervised Object-oriented Scene Representation via Spatial Attention and Decomposition." In: *ArXiv preprint arXiv:2001.02407*.

Locatello, Francesco, Dirk Weissenborn, Thomas Unterthiner, Aravindh Mahendran, Georg Heigold, Jakob Uszkoreit, Alexey Dosovitskiy, and Thomas Kipf (2020). "Object-centric learning with slot attention." In: *Advances in Neural Information Processing Systems*.

Ma, Xuezhe and Eduard Hovy (2016). "End-to-end Sequence Labeling via Bi-directional LSTM-CNNs-CRF." In: *Annual Meeting of the Association for Computational Linguistics (ACL)*.

Namysl, Marcin, Sven Behnke, and Joachim Köhler (2020). "NAT: Noise-aware Training for Robust Neural Sequence Labeling." In: *Annual Meeting of the Association for Computational Linguistics (ACL)*.

Olah, Christopher (2015). *Understanding LSTM Networks.* URL: `http://colah.github.io/posts/2015-08-%7BU%7Dnderstanding-%7BL%7D%7BS%7D%7BT%7D%7BM%7Ds/`.

Oprea, Sergiu, Pablo Martinez-Gonzalez, Alberto Garcia-Garcia, John Alejandro Castro-Vargas, Sergio Orts-Escolano, Jose Garcia-Rodriguez, and Antonis Argyros (2020). "A review on deep learning techniques for video prediction." In: *IEEE Transactions on Pattern Analysis and Machine Intelligence.*

Pascanu, Razvan, Tomas Mikolov, and Yoshua Bengio (2013). "On the Difficulty of Training Recurrent Neural Networks." In: *International conference on machine learning.* PMLR.

Periyasamy, Arul Selvam, Max Schwarz, and Sven Behnke (2021). "Synpick: A dataset for dynamic bin picking scene understanding." In: *2021 IEEE 17th International Conference on Automation Science and Engineering (CASE).* IEEE.

Poibrenski, Atanas, Matthias Klusch, Igor Vozniak, and Christian Müller (2020). "M2p3: Multimodal Multi-pedestrian Path Prediction by Self-driving Cars with Egocentric Vision." In: *Proceedings of the 35th Annual ACM Symposium on Applied Computing.*

Ronneberger, Olaf, Philipp Fischer, and Thomas Brox (2015). "U-net: Convolutional Networks for Biomedical Image Segmentation." In: *International Conference on Medical Image Computing and Computer-Assisted Intervention.* Springer.

Samek, Wojciech, Thomas Wiegand, and Klaus-Robert Müller (2017). "Explainable Artificial Intelligence: Understanding, Visualizing and Interpreting Deep Learning Models." In: *ITU Journal: ICT Discoveries.*

Santana, Eder and George Hotz (2016). "Learning a Driving Simulator." In: *ArXiv preprint arXiv:1608.01230.*

Shi, Xingjian, Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai-Kin Wong, and Wang-chun Woo (2015). "Convolutional LSTM network: A Machine Learning Approach for Precipitation Nowcasting." In: *Advances in neural information processing systems.*

Simonyan, Karen and Andrew Zisserman (2015). "Very Deep Convolutional Networks for Large-scale Image Recognition." In: *International Conference on Learning Representations (ICLR).*

Spelke, Elizabeth S and Katherine D Kinzler (2007). "Core knowledge." In: *Developmental science.*

Szegedy, Christian, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich (2015). "Going Deeper with Convolutions." In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.*

Tay, Yi, Mostafa Dehghani, Dara Bahri, and Donald Metzler (2020). "Efficient transformers: A survey." In: *ACM Computing Surveys (CSUR).*

Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin (2017). "Attention is all you need." In: *Advances in Neural Information Processing Systems.*

Villar-Corrales, Angel and Sven Behnke (2021). “Unsupervised Image Decomposition with Phase-Correlation Networks.” In: *International Conference on Computer Vision Theory and Applications.*

Vinyals, Oriol, Alexander Toshev, Samy Bengio, and Dumitru Erhan (2016). “Show and Tell: Lessons Learned from the 2015 MSCOCO Image Captioning Challenge.” In: *IEEE transactions on pattern analysis and machine intelligence.*

Wang, Jiangliu, Jianbo Jiao, and Yun-Hui Liu (2020). “Self-supervised Video Representation Learning by Pace Prediction.” In: *European conference on computer vision.* Springer.

Wang, Yunbo, Mingsheng Long, Jianmin Wang, Zhifeng Gao, and Philip S Yu (2017). “Predrnn: Recurrent Neural Networks for Predictive Learning Using Spatiotemporal LSTMs.” In: *Advances in Neural Information Processing Systems.*

Wang, Zhou, A.C. Bovik, H.R. Sheikh, and E.P. Simoncelli (2004). “Image quality assessment: from error visibility to structural similarity.” In: *IEEE Transactions on Image Processing.*

Watters, Nicholas, Loic Matthey, Christopher P Burgess, and Alexander Lerchner (2019). “Spatial broadcast decoder: A Simple Architecture for Learning Disentangled Representations in VAEs.” In: *ArXiv preprint arXiv:1901.07017.*

Weis, Marissa A, Kashyap Chitta, Yash Sharma, Wieland Brendel, Matthias Bethge, Andreas Geiger, and Alexander S Ecker (2020). “Unmasking the Inductive Biases of Unsupervised Object Representations for Video Sequences.” In: *ArXiv preprint arXiv:2006.07034.*

Wu, Huikai, Junge Zhang, Kaiqi Huang, Kongming Liang, and Yizhou Yu (2019). “Fastfcn: Rethinking Dilated Convolution in the Backbone for Semantic Segmentation.” In: *ArXiv preprint arXiv:1903.11816.*

Wu, Yi-Fu, Jaesik Yoon, and Sungjin Ahn (2021). “Generative Video Transformer: Can Objects be the Words?” In: *International Conference on Machine Learning.* PMLR.

Yan, Wilson, Ryo Okumura, Stephen James, and Pieter Abbeel (2022). “Patch-based Object-centric Transformers for Efficient Video Generation.” In: *ArXiv preprint arXiv:2206.04003.*

Ye, Yufei, Maneesh Singh, Abhinav Gupta, and Shubham Tulsiani (2019). “Compositional Video Prediction.” In: *Proceedings of the IEEE/CVF International Conference on Computer Vision.*

Zang, Xiao, Miao Yin, Lingyi Huang, Jingjin Yu, Saman Zonouz, and Bo Yuan (2022). “Robot Motion Planning as Video Prediction: A Spatio-Temporal Neural Network-based Motion Planner.” In: *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS).* IEEE.

Zhang, Aston, Zachary C Lipton, Mu Li, and Alexander J Smola (2021). “Dive into deep learning.” In: *ArXiv preprint arXiv:2106.11342.*

Zhang, Richard, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang (2018). “The Unreasonable Effectiveness of Deep Features as a Perceptual Metric." In: *Proceedings of the IEEE conference on computer vision and pattern recognition.*