Rheinische
Friedrich-Wilhelms-
Universität Bonn

Institute for Computer Science
Department VI
Autonomous Intelligent Systems

Rheinische Friedrich-Wilhelms-Universität Bonn

Master Thesis

Object-Centric Image to Video Generation with Language Guidance

Author:
Gjergj Plepi

First Examiner:
Prof. Dr. Sven Behnke

Second Examiner:
Prof. Dr. Jürgen Gall

Supervisor:
M. Sc. Angel Villar-Corrales

Date: January 31, 2025

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

Accurate and robust world models are essential for autonomous systems to understand their environment and predict future outcomes. Recent advancements in object-centric models have demonstrated strong capabilities in modeling object dynamics and interactions through structured latent spaces. However, existing approaches face limitations in handling datasets with complex object dynamics and lack the ability to incorporate external guidance, reducing their effectiveness as world models in robotics applications. In this work, we introduce TextOCVP, a novel object-centric model for image-to-video generation with language guidance. TextOCVP extracts object representations, or slots, from a given scene, and employs a text-conditioned Transformer-based predictor to forecast future object states and generate video frames. By integrating textual guidance and jointly modeling object interactions and dynamics, TextOCVP enables accurate, interpretable, and controllable predictions. We evaluate our approach on various datasets, and demonstrate its superiority over existing text-conditioned image-to-video generative models that do not leverage object-centricity. To the best of our knowledge, TextOCVP is the first model to combine object-centric representation learning with text-conditioned video generation, paving the way for future research in this area.

Contents

1 Introduction

Understanding and reasoning about the real-world environment is essential for enabling autonomous systems to better comprehend their surroundings, predict future events, and adapt their actions accordingly. Humans naturally perceive their environment as a structured composition of individual objects that interact and evolve dynamically over time. Inspired by this, neural networks equipped with compositional inductive biases have shown the ability to learn structured, object-centric representations of the world. This capability has given rise to the field of object-centric representation learning, which focuses on describing individual objects within a scene by leveraging low-level perceptual features.

In recent years, significant progress has been made in unsupervised object-centric representation learning. Early approaches focused on extracting object representations from simple, synthetic images (Burgess et al. 2019, Lin et al. 2020, Locatello et al. 2020). Subsequent methods, such as SAVi (Kipf et al. 2022) and SAVi++ (Elsayed et al. 2022), extended these techniques to videos. More recently, breakthroughs like the work of Seitzer et al. 2023 have demonstrated the scalability of object-centric models to real-world scenes.

These advancements have enabled various applications in future prediction and planning by enabling modeling scene dynamics at the object level. Particularly, models like OCVP (Villar-Corrales, Wahdan, and Behnke 2023) and SlotFormer (Z. Wu et al. 2023) introduced object-centric world models that explicitly captured the spatio-temporal relationships between objects in a scene. This marked a shift from earlier works that primarily focused on modeling temporal dynamics at the image level, ignoring the compositional structure of the environment (Y. Wang et al. 2023, Villar-Corrales, Karapetyan, et al. 2022, Rakhimov et al. 2021, Gao et al. 2022).

Despite these advancements, current approaches face challenges when applied to datasets with complex object dynamics and fail to incorporate external guidance. These limitations restrain their effectiveness as world models, particularly in robotic applications. To address these issues, we propose TextOCVP, a novel object-centric model for image-to-video generation with language guidance. Our model enables the forecasting of future scene dynamics based on a reference image and textual instructions. TextOCVP extracts object representations from

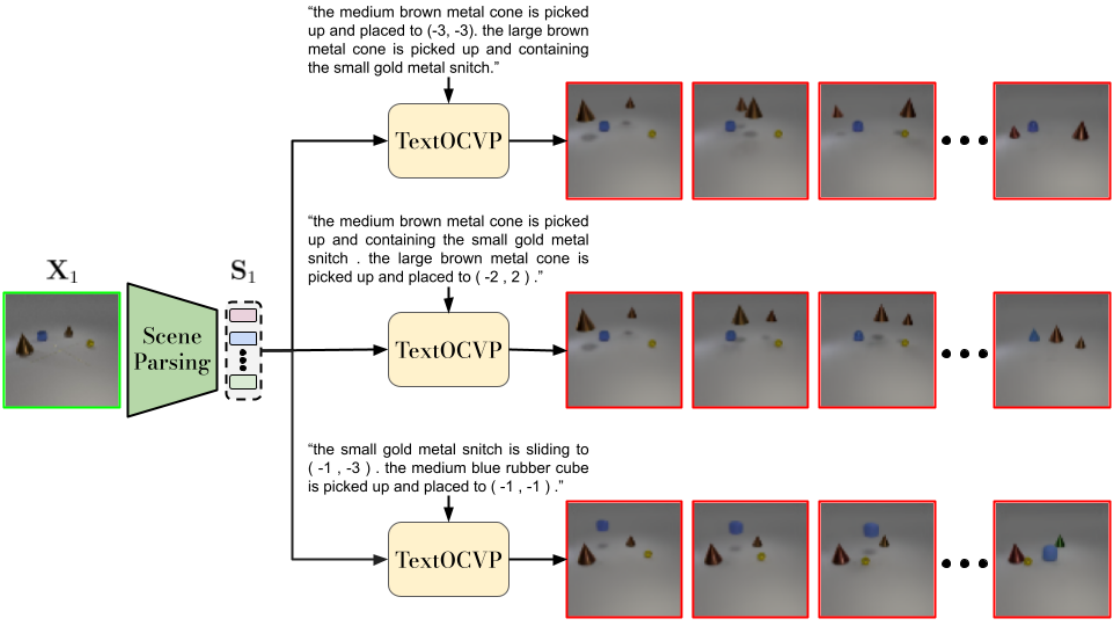


Figure 1.1: An overview of TextOCVP.

the input image and predicts future dynamics using a text-conditioned predictor. By jointly modeling spatio-temporal object relationships and integrating textual guidance, it can generate future object states and video frames that align with the given language instructions, as depicted in Figure 1.1. We introduce a new predictor architecture that incorporates text information through text-to-slot attention modules, and adapt a new architecture for learning object-centric representations, which is leveraged to parse scenes into object representations and render future video frames from predicted object states.

To evaluate our approach, we conduct extensive experiments on the CATER (Girdhar and Ramanan 2020) and CLIPort (Shridhar, Manuelli, and Fox 2021) datasets. First, we assess our model on the video prediction task using both CATER-easy and CATER-hard variants, emphasizing the benefits of textual information in improving future frame prediction. Our results, both quantitative and qualitative, demonstrate the superiority of our model over existing text-conditioned methods that do not leverage object-centricity, such as MAGE (Hu, Luo, and Chen 2022) and SEER (Gu et al. 2024).

Additionally, we highlight the importance of our proposed object-centric module in learning representations for complex scenes. Through an extensive model analysis, we show that TextOCVP can adapt its generated video sequences based on language instructions, effectively aligning predicted motions and object states with the provided descriptions. Visualizations further demonstrate how critical text in-

formation influences model predictions. To the best of our knowledge, TextOCVP is the first model to combine text-conditioned and object-centric approaches for video generation, setting a new direction for future research in this area.

In summary, our main contributions on this work are:

1. We introduce TextOCVP, a novel object-centric model for text-driven image-to-video generation.

2. We propose a new text-conditioned predictor and object-centric module, enabling controllability over model-generated sequences and scaling to more complex datasets.

3. Through extensive evaluation, we demonstrate that TextOCVP outperforms existing text-conditioned approaches, both quantitatively and qualitatively, by leveraging object-centric representations.

1.1 Thesis Overview

The thesis is structured as follows:

Chapter 2. This chapter introduces the foundational concepts and technologies used to build our model. We provide an overview of Feed-Forward Networks, Transformers, Convolutional Neural Networks (CNNs), and Vision Transformers (ViT), which form the building blocks of our architecture.

Chapter 3. We review related works that informed and inspired our approach. This includes advancements in object-centric learning and the evolution of object-centric models for video prediction. We also examine concurrent methods for text-driven image-to-video generation, identifying the lack of object-centricity in existing approaches.

Chapter 4. This chapter provides a comprehensive description of our proposed model, TextOCVP. We detail its architecture, highlighting the novel text-conditioned predictor and object-centric module, which enable controllability and scalability. We also explain the training and inference procedures and provide implementation details of different model variants.

Chapter 5. We describe the experimental setup, including datasets and evaluation metrics used to assess performance. We conduct quantitative and qualitative evaluations that demonstrate the advantages of TextOCVP over existing methods. Additionally, we present ablation studies, analyze model behavior depending on the language instructions, and illustrate how textual information enhances predictions.

Chapter 6. Finally, we summarize our approach, findings, and contributions. We discuss the limitations of our work and outline future research directions to further advance object-centric and text-driven image-to-video generation.

2 Fundamentals

2.1 Feed-Forward Networks

The development of neural networks was inspired by biological systems, particularly the functioning of the human brain in processing information. This aspiration led to the creation of key elements of neural networks, including interconnected processing units (neurons), adjustable weights linking these neurons, nonlinear activation functions, and the concept of mapping an input vector to an output value. Collectively, these components form the foundation of neural networks as systems of simple information-processing units interconnected by directed, weighted connections.

The model of an artificial neuron is depicted in Figure 2.1, illustrating its input signals, weighted connections, weighted sum, nonlinear activation function, and scalar output. Mathematically, neuron k can be described by the following equations (Haykin 1998):

$$v_k = \sum_{i=1}^{m} x_i w_{ki} + b_k, \tag{2.1}$$

and

$$y_k = \varphi(v_k). \tag{2.2}$$

where $x_1, x_2, \ldots, x_m$ are the input signals; $w_{k1}, w_{k2}, \ldots, w_{km}$ are the weights of neuron k; v_k represents the weighted sum of inputs; b_k denotes the bias term; $\varphi(.)$ is the nonlinear activation function; and y_k is the neuron's output. The choice of the activation function φ depends on the task to be addressed. Some commonly used activation functions are summarized in Table 2.1.

Feed-forward networks (FFNs) are neural networks where neurons are structured into an input layer, one or more hidden processing layers, and an output layer. Hidden layers are crucial for extracting higher-order features from input signals (Haykin 1998). In these networks, neurons are connected in an unidirectional manner, with signals flowing only towards the neurons of the next layer. When

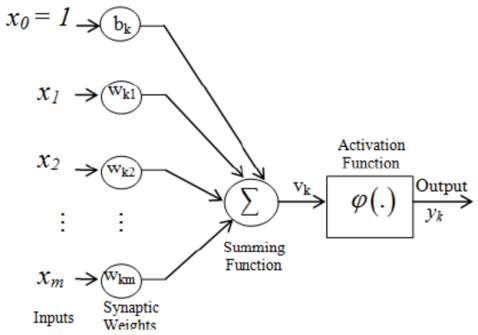


Figure 2.1: Nonlinear model of a neuron, Haykin 1998

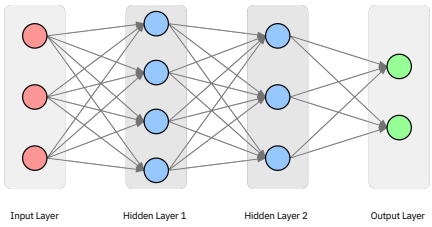


Figure 2.2: Fully-connected feed-forward network structure.

Table 2.1: Common nonlinear activation functions.

Activation Functions	
Sigmoid	$\varphi(x) = \frac{1}{1+e^{-x}}$
Hyperbolic tangent (tanh)	$\varphi(x) = \tanh(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$
Rectified linear unit (ReLU)	$\varphi(x) = \begin{cases} 0, & \text{for } x \leq 0 \\ x, & \text{for } x > 0 \end{cases}$
Gaussian Error Linear Unit (GELU)	$\varphi(x) = x \cdot \Phi(x) = x \cdot \frac{1}{2}\left[1 + \operatorname{erf}\left(\frac{x}{\sqrt{2}}\right)\right]$
SoftPlus	$\varphi(x) = \ln(1 + e^{x})$
Softmax	$\varphi(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{n} e^{x_j}}$

all neurons in one layer are connected to every neuron in the next, the network is referred to as fully-connected, as shown in Figure 2.2.

It has been proven that feed-forward networks with a single hidden layer containing a finite number of neurons and nonlinear activation functions can approximate any continuous function mapping from an N-dimensional space to an M-dimensional space to an arbitrary degree of accuracy. This universal approximation property (Cybenko 1989, Hornik, Stinchcombe, and White 1989) highlights the expressive power of these networks. However, the theorem does not provide guidance on the number of neurons required or the process of determining appropriate weights. This leads to the process of model training and learning.

Model training involves adjusting the network's weights to minimize an error function. This optimization process depends on a learning algorithm and training data. Learning approaches are typically categorized into supervised, unsupervised, and reinforcement learning.

In unsupervised learning, the network identifies patterns in the input data without labeled outputs, classifying similar data into groups. Whereas, in supervised learning, training data includes both inputs and desired outputs, allowing error computation by comparing actual and predicted outputs. Model parameters are adjusted iteratively to minimize this error, typically using optimization methods such as gradient descent. The backpropagation algorithm computes gradients of the error function with respect to weights, which are then leveraged by the optimizer to update the network's weights.

2.2 Convolutional Neural Networks (CNNs)

Convolutional Neural Networks (CNNs) (LeCun, Boser, et al. 1989) represent a specialized class of neural networks designed to process data with an inherent grid-like structure, such as images, which can be thought as two-dimensional grids of pixels (Goodfellow 2016). By leveraging the structured nature of the input, CNN architectures embed specific properties that enhance computational efficiency and significantly reduce the number of parameters required within the network.

2.2.1 CNNs for Image Processing

Traditional feed-forward neural networks, as described in Section 2.1, are not well-suited for processing full-scale images. Images typically consist of hundreds or even thousands of pixels, which correspond to a large number of input variables. The direct use of such high-dimensional inputs in fully-connected neural networks results in an excessive number of parameters, increasing the risk of overfitting. Moreover, these networks lack inherent mechanisms to handle translations or local distortions in the input and do not account for the topological structure of the data.

In contrast, CNNs arrange neurons in three dimensions: width, height, and depth, accommodating the additional depth dimension necessary for image processing. Images have strong local 2D structures, where adjacent pixels are often highly correlated (LeCun, Bengio, et al. 1995). CNNs exploit this property by constraining the receptive fields of hidden units to be local, meaning that neurons in a given layer are only connected to a small region of the preceding layer. This localized connectivity facilitates the extraction of spatially relevant features. These local features are further combined in higher layers of the network, enabling a hierarchical learning of features.

Additionally, CNNs implement parameter sharing, where the same set of weights is shared across neurons within the same depth level, and applied across different

spatial locations of the input through convolutional operation. This approach not only reduces the number of parameters but also introduces translational invariance, enabling the network to recognize patterns regardless of their position within the input.

2.2.2 Layers of CNNs

A CNN comprises a sequence of layers, each transforming an input volume of activations into an output volume through a differentiable function. CNNs are constructed by combining three primary types of layers: Convolutional Layers, Pooling Layers, and Fully Connected Layers (discussed in Section 2.1). Among these, the Pooling Layer does not involve learnable parameters, whereas the other two do.

Convolutional Layer. The Convolutional Layer serves as the fundamental building block of a CNN. Its parameters consist of learnable filters (also referred to as kernels), which are small in spatial dimensions (width and height) but span the entire depth of the input volume. During the forward pass, each filter is convolved across the spatial dimensions of the input, computing dot products between the filter's weights and the input values at each position. This operation produces a 2D output known as an activation map, representing the filter's response at different spatial locations.

These filters learn to detect specific visual features, such as edges and corners in the initial layers, and more complex patterns or entire objects in deeper layers, i.e. hierarchical feature learning. The activation maps generated by all filters are stacked along the depth dimension, forming the output volume.

The spatial size of the filters (or receptive fields) is a critical hyperparameter, typically set to dimensions such as 3×3, 5×5, or 7×7. CNNs leverage parameter sharing, meaning all neurons within a 2D slice of depth use the same filter parameters. This design reduces the total number of parameters and aligns with the intuition that features useful at one spatial location should be equally valuable at other locations in the input.

The number of filters determines the depth of the output volume, with each filter capturing distinct features. The spatial dimensions of the output activation map depend on several hyperparameters: stride (the step size for moving the filter across the input), dilation (which allows skipping pixels for wider receptive fields with fewer parameters), and padding (the addition of zero-valued pixels around the input's borders). Zero-padding not only addresses sizing inconsistencies but also preserves edge information.

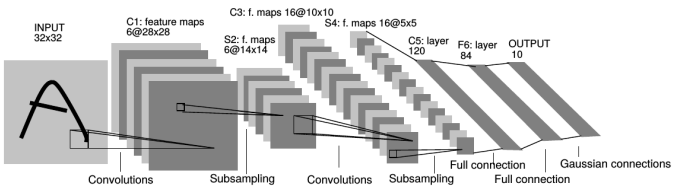


Figure 2.3: The LeNet-5 architecture, LeCun, Bottou, et al. 1998

The spatial dimensions of the output volume $(H_{\text{out}}, W_{\text{out}})$ can be calculated as a function of the input dimensions $(H_{\text{in}}, W_{\text{in}})$, filter size (F), stride (S), dilation factor (D), and padding (P), as shown below:

$$\begin{cases} H_{\text{out}} = \frac{H_{\text{in}}+2P-D\cdot(F-1)-1}{S} + 1, \\ W_{\text{out}} = \frac{W_{\text{in}}+2P-D\cdot(F-1)-1}{S} + 1. \end{cases} \tag{2.3}$$

Pooling Layer. Pooling Layers are commonly inserted between Convolutional Layers in a CNN architecture. Their primary function is to reduce the spatial dimensions of the feature representations, thereby lowering the computational complexity and the risk of overfitting. Pooling operations are applied independently to each depth slice of the input, using functions such as maximum (Max Pooling) or average (Mean Pooling) over a specified receptive field. These operations are parameter-free, as they compute fixed transformations of the input.

2.2.3 CNN Architectures

Numerous CNN architectures have been developed, varying in how they combine the three primary layers described above and in the inclusion of additional components such as activation functions (e.g., ReLU), normalization layers, or dropout layers.

One of the earliest successful applications of CNNs was LeNet (LeCun, Bottou, et al. 1998), which inspired subsequent architectures. It featured convolutional layers for feature extraction, average pooling for subsampling, and a Multi-Layer Perceptron (MLP) as the classifier, as illustrated in Figure 2.3. An important architecture was AlexNet (Krizhevsky, Sutskever, and Geoffrey E Hinton 2012), which introduced a deeper and larger network architecture with stacked convolutional layers. Unlike earlier designs, which typically alternated between a single convolutional layer and a pooling layer, AlexNet utilized multiple consecutive convolutional layers, enabling it to learn more complex features.

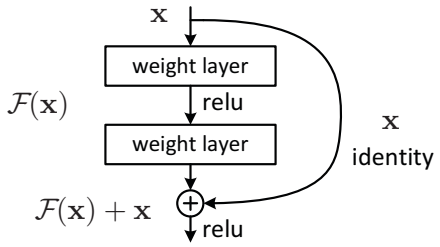


Figure 2.4: Residual block used in ResNet, He et al. 2016

ZFNet (Zeiler 2014) improved upon AlexNet by fine-tuning hyperparameters, such as reducing the stride and filter size in the initial layers and increasing the size of intermediate layers. This architecture also provided insights into the features learned by CNNs. VGGNet (Simonyan and Zisserman 2015) demonstrated the importance of network depth for improved performance, employing small filters (3×3) and deep architectures. The Residual Network (ResNet) by He et al. 2016 introduced skip connections, as depicted in Figure 2.4, which allowed gradients to flow directly through deeper networks, addressing the vanishing gradient problem. ResNet also extensively utilized batch normalization and omitted fully connected layers at the network's end. This architecture achieved significant performance gains over its predecessors.

2.3 Transformer

A sequence-to-sequence model processes an input sequence of items and outputs a corresponding sequence. Traditionally, these models utilized an encoder-decoder architecture based on Recurrent Neural Networks (RNNs) (Rumelhart, Geoffrey E Hinton, and Williams 1986), enhanced with attention mechanisms (Bahdanau, Cho, and Bengio 2015) to focus on relevant parts of the input, thus enabling the processing of longer sequences. Vaswani et al. 2017 introduced the Transformer network as a sequence-to-sequence model that does not use any recurrent networks, relying only on attention mechanisms. This design enables efficient parallelization during computations, resulting in faster training and significant performance improvements.

2.3.1 Encoder-Decoder Architecture

The Transformer network employs an encoder-decoder architecture. The encoder maps the input sequence $x = (x_1, ..., x_n)$ to a sequence of continuous representa-

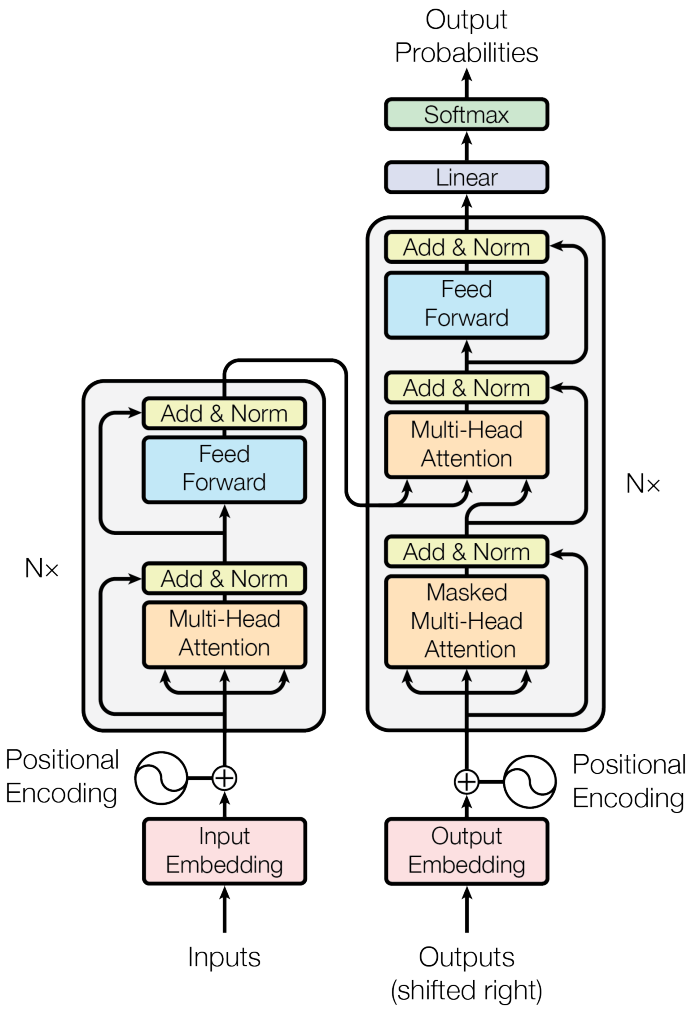


Figure 2.5: The Transformer network architecture, Vaswani et al. 2017

tions $z = (z_1, ..., z_n)$. Given z, the decoder outputs a sequence $y = (y_1, ..., y_m)$ in an auto-regressive manner. This means that at each time step, the decoder produces a single element, which is then used as an additional input in next time steps. Both the encoder and decoder in the Transformer model consists of stacked layers of self-attention mechanisms and point-wise, fully-connected feed-forward networks, as illustrated in Figure 2.5.

The encoder is composed of six identical layers, each containing two sub-layers. The first sub-layer implements a multi-head self-attention mechanism, while the second sub-layer is a fully-connected feed-forward network applied independently at each position of the output from the attention sub-layer. Both sub-layers employ residual connections (He et al. 2016), and are followed by layer normalization (L. J.

Ba, Kiros, and Geoffrey E. Hinton 2016). The dimensionality of the input and output for each sub-layer is $d_{\text{model}} = 512$.

The decoder shares a similar architecture with the encoder, consisting of six identical layers that include the two sub-layers of the encoder. However, each decoder layer contains an additional sub-layer, named cross-attention, which performs multi-head attention over the encoder's output. As in the encoder, each sub-layer in the decoder is equipped with residual connections and layer normalization. Moreover, the self-attention mechanism in the decoder is modified through masking to ensure that each position attends only to previous positions in the sequence.

2.3.2 Attention in Transformers

The attention mechanism was introduced by Bahdanau, Cho, and Bengio 2015 to improve the capability of sequence-to-sequence models to effectively process and retain information from long input sequences. It can be broadly interpreted as a vector of importance weights or scores that determine the degree of alignment between a predicted element and the input elements. This mechanism enables the model to focus selectively on relevant parts of the input sequence. Various attention mechanisms have been proposed, including additive attention by Bahdanau, Cho, and Bengio 2015 and dot-product attention by Luong, Pham, and Manning 2015, which primarily differ in the design of their alignment score functions. When the attention mechanism relates different positions of the same input sequence, it is called self-attention (Cheng, Dong, and Lapata 2016).

The attention function in the Transformer model is formulated as a mapping of a query and a set of key-value pairs to an output, with the query, keys, values, and output all represented as vectors. For a given position, the output is computed as a weighted sum of the values, where the weights are derived using a compatibility function that measures the alignment between the query and each corresponding key. The attention mechanism used in Transformer is called Scaled Dot-Product Attention, illustrated in Figure 2.6.

Given an input sequence, each input element is first embedded and then transformed into corresponding query, key, and value vectors by multiplication with learned matrices, $W^Q \in \mathbb{R}^{d_{\text{model}} \times d_k}$, $W^K \in \mathbb{R}^{d_{\text{model}} \times d_k}$, and $W^V \in \mathbb{R}^{d_{\text{model}} \times d_v}$, respectively, where d_k is the dimension of the query and key vectors, while d_v is the dimension of the value vectors. To compute the output for a specific position, the scores are calculated as a dot product of its query vector with all keys, each divided by $\sqrt{d_k}$, and then a softmax function is applied to generate the attention weights. These weights are used to compute the output as a weighted sum of the

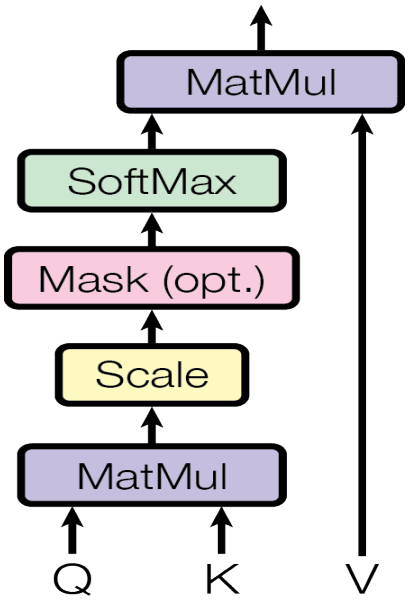


Figure 2.6: Scaled Dot-Product Attention, Vaswani et al. 2017

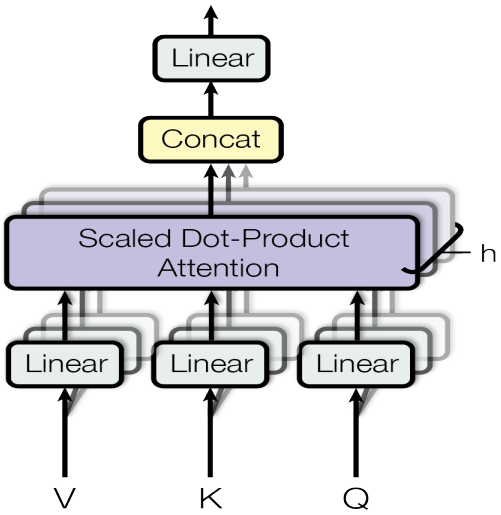


Figure 2.7: Multi-Head Self-Attention, Vaswani et al. 2017

value vectors. The division by $\sqrt{d_k}$ helps in gradient stability, especially for large values of d_k.

In practice, computations are performed simultaneously in parallel for all queries, with the query, key, and value vectors stacked into matrices Q, K, and V. The outputs are then computed efficiently using the following equation:

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V. \tag{2.4}$$

In the encoder-decoder attention sub-layer of the decoder, the key and value matrices (K and V) are derived from the encoder's output, allowing the decoder to selectively focus on relevant parts of the input sequence. This attention mechanism is called cross-attention, since a sequence can attend to another sequence's information. In the self-attention sub-layer in both the encoder and decoder, all of the keys, values and queries are computed from the output of the previous layer (or directly from the input sequence in the first layer). Within the encoder, every position in the sequence can attend to all other positions, while the decoder restricts attention to positions up to and including the current position to maintain the auto-regressive property. This is implemented by masking future positions, setting their attention score to $-\infty$ before the softmax operation, a process known as masked self-attention.

2.3.3 Multi-Head Attention

Elements within a sequence relate to each other in various ways, motivating the extension of the attention mechanism to multiple heads, resulting in the multi-head attention mechanism. The Transformer employs $h = 8$ heads, where each head corresponds to an independent attention layer with its own set of linear projection matrices. Specifically, the queries, keys, and values are projected h times using different learned projection matrices. The attention function is then applied in parallel to these independently projected queries, keys and values, producing d_v-dimensional output vectors for each head. These outputs are then concatenated and passed through a linear projection to produce the final output, as illustrated in Figure 2.7. The computation of multi-head attention can be expressed as:

$$\begin{aligned} \text{MultiHead}(Q, K, V) &= \text{Concat}(\text{head}_1, ..., \text{head}_h)W^O, \\ \text{head}_i &= \text{Attention}(QW_i^Q, KW_i^K, VW_i^V), \end{aligned} \tag{2.5}$$

where the projections matrices $W_i^Q \in \mathbb{R}^{d_{\text{model}} \times d_k}$, $W_i^K \in \mathbb{R}^{d_{\text{model}} \times d_k}$, $W_i^V \in \mathbb{R}^{d_{\text{model}} \times d_v}$ and $W^O \in \mathbb{R}^{h*d_v \times d_{\text{model}}}$ are learnable parameters of the model, and $d_k = d_v = d_{\text{model}}/h = 64$. The multi-head attention mechanism enables each head to capture distinct aspects of the relationships between input elements. This allows the model to attend to diverse information from different representation subspaces, enhancing its ability to model complex dependencies within the input sequence.

2.3.4 Position-wise Feed-Forward Network

In addition to the attention sub-layers, the encoder and decoder also include a sub-layer comprising a fully connected feed-forward network. This network consists of two linear layers with a ReLU activation function applied between them, as described in Equation (2.6). The feed-forward network operates independently at each position in the sequence, applying the same transformation across all positions. However, the parameters of the feed-forward network differ between layers.

$$\text{FFN}(x) = \max(0, xW_1 + b_1)W_2 + b_2. \tag{2.6}$$

2.3.5 Inputs and Outputs

Transformer employs learned embeddings, stored in an embedding matrix, to convert input and output tokens into vectors of dimension d_{model}. The encoder module

processes the input sequence and outputs a representation with the same shape as the input, but with each position enriched by attending to other positions in the sequence. On the decoder side, the output is a vector of continuous values that needs to be converted into next-token probabilities. To achieve this, a linear layer maps the vector to a logits vector with dimensionality equal to the vocabulary size. Subsequently, a softmax function is applied to convert the logits into probabilities, as shown in Figure 2.5.

2.3.6 Positional Encoding

Recurrent Neural Networks (RNNs) process tokens sequentially, inherently capturing information about the order of the sequence. In contrast, the Transformer relies solely on attention mechanisms and has no information about the sequence order. To address this issue, the Transformer incorporates positional encodings by adding a unique vector to each input embedding. These encodings provide information about the absolute or relative position of tokens in the sequence, enabling the model to identify the position of each token or extract information regarding the distance between different tokens in the sequence. The positional encodings used in Transformer are derived from sinusoidal functions, as defined below:

$$\begin{aligned} \mathrm{PE}(\mathrm{pos}, 2i) &= \sin(\mathrm{pos}/10000^{2i/d_{\mathrm{model}}}), \\ \mathrm{PE}(\mathrm{pos}, 2i+1) &= \cos(\mathrm{pos}/10000^{2i/d_{\mathrm{model}}}), \end{aligned} \tag{2.7}$$

where pos is the position and i is the dimension. Each dimension of the positional encoding corresponds to a sinusoidal wave. This formulation facilitates the model's ability to learn relative positions and generalize to sequences longer than those encountered during training (Vaswani et al. 2017).

2.4 Vision Transformer (ViT)

The original Transformer architecture (Vaswani et al. 2017) was designed to address machine translation tasks by processing sequences of words. Its advantages over Recurrent Neural Networks (RNNs), including superior performance, parallelized computations, and the ability to capture long-term dependencies, made it the preferred model in the field of Natural Language Processing (NLP). Inspired by this success in NLP, the Vision Transformer (ViT) was introduced by Kolesnikov et al. 2021 as the first pure Transformer model applied directly to images. ViT is trained for image classification in a supervised manner.

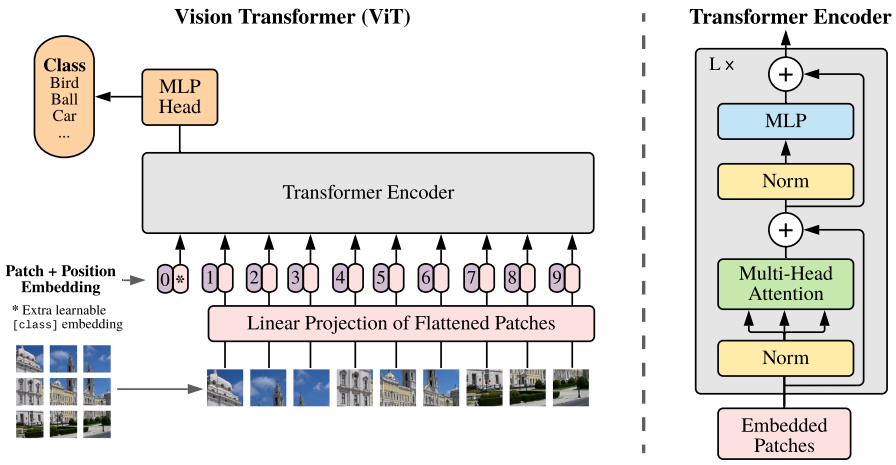


Figure 2.8: ViT model overview, Kolesnikov et al. 2021

2.4.1 Model Overview

In ViT, the input image is first split into fixed-size, non-overlapping patches, which act as a sequence of tokens. Each patch is flattened and linearly projected into a corresponding patch embedding. A learnable class token embedding is prepended to the sequence of embedded patches and learnable 1D position embeddings are added to retain positional information.

The resulting sequence is fed into a standard Transformer encoder, as described in Section 2.3.1. The encoder processes the sequence and outputs representations for each position. The representation corresponding to the prepended class token is then passed to a feed-forward neural network, followed by a softmax layer, to predict the final image class. The overall process is illustrated in Figure 2.8.

2.4.2 Scaling and Inductive Biases

ViT adopts a training strategy inspired by NLP models, such as BERT (Devlin et al. 2019), which involves pre-training on large-scale datasets followed by fine-tuning on smaller target datasets for downstream tasks. ViT is fine-tuned with higher-resolution images than those used during pre-training, as it is often show that it is more beneficial (Touvron et al. 2019), but the patch size remains constant, resulting in longer sequences. ViT can handle arbitrary sequence lengths, but the pre-trained position embeddings need to be 2D-interpolated to align with the new resolution and patch locations.

Unlike Convolutional Neural Networks (CNNs), ViT incorporates significantly less image-specific inductive bias. The resolution adjustment and patch extraction are the only points where prior knowledge about the 2D structure of the images is

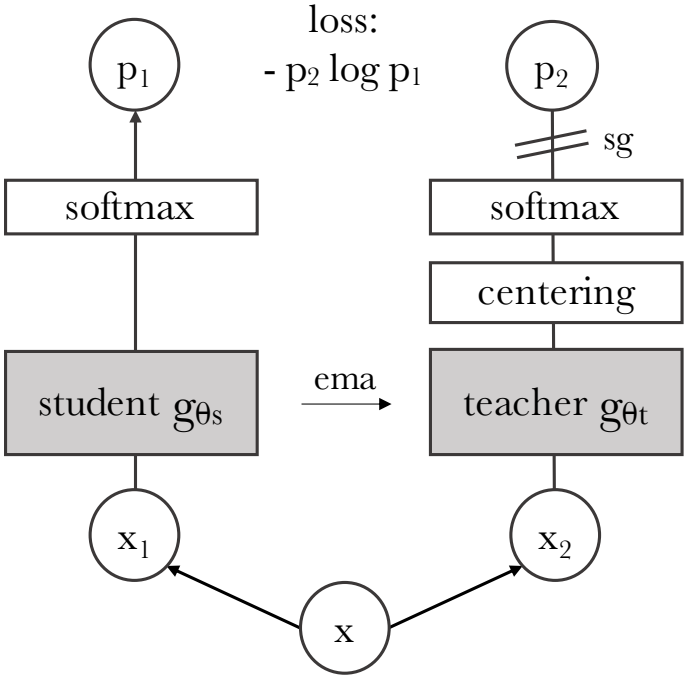


Figure 2.9: DINO framework using a single pair of views (x_1, x_2), Caron, Touvron, et al. 2021

manually incorporated into the Vision Transformer. Initially, the position embeddings have no inherent information about the 2D positions of the patches, requiring the model to learn all spatial relations from scratch. Furthermore, only the MLP layers are local and translationally equivariant, while the self-attention layers are global.

2.4.3 Towards Self-Supervised Learning

While ViTs achieve competitive performance compared to CNNs, they are often more computationally demanding, require larger datasets for training, and their features do not exhibit unique properties. A major factor contributing to the success of Transformers in NLP is large-scale self-supervised pre-training (Devlin et al. 2019), and the use of supervision in pre-training can be a limiting factor for ViT. Image-level supervision often reduces the rich visual information contained in images to a single concept from a predefined set of object categories (Russakovsky et al. 2015).

DINO, introduced by Caron, Touvron, et al. 2021, and its successor DINOv2 (Oquab et al. 2023), introduce a self-supervised learning approach for pre-training Vision Transformers. DINO is intepreted as a form of self-distillation with no

labels, where a student network learns to match the outputs of a teacher network. Both networks share the same architecture, consisting of a ViT backbone and a projection head, but their parameters differ.

In this framework, given an image, a set of distorted image views (or crops) is generated using a multi-crop strategy (Caron, Misra, et al. 2020). These views are passed through the student network, while only the global views are processed by the teacher network, enforcing local-to-global correspondences. The similarity between the features produced by the two networks is measured using a cross-entropy loss. The teacher network is built from past iterations of the student network by applying an exponential moving average (EMA) to the student weights. It remains fixed during training, with gradients propagated only through the student. This process is illustrated in Figure 2.9.

The features produced by ViTs pre-trained with DINO, known as DINO ViTs, are used as features in downstream tasks, such as object detection and segmentation. Notably, these features explicitly capture scene layouts and object boundaries. Caron, Touvron, et al. 2021 also highlight the benefit of using smaller patch sizes in ViTs to improve the quality of the learned features.

3 Related Work

3.1 Object-Centric Representation Learning

Representation learning has emerged as a critical area in machine learning, as the ability to derive meaningful representations from data often enhances model performance by improving its comprehension of the input space (Bengio, Courville, and Vincent 2013). Among various approaches, object-centric representation learning focuses on simultaneously learning representations that characterize individual objects within an input image or video.

In recent years, numerous methods have been proposed for unsupervised object-centric decomposition of images, including IODINE (Greff et al. 2019), MONet (Burgess et al. 2019), SPACE (Lin et al. 2020), PCDNet (Villar-Corrales and Behnke 2022), and Slot Attention (Locatello et al. 2020).

The Slot Attention module maps a set of input feature vectors to a corresponding set of output vectors referred to as *slots*. Each slot can describe and bind to any object in the input, enabling the module to generalize to novel compositions, handle varying numbers of objects, and dynamically allocate slots. Slot Attention employs an iterative attention mechanism to assign parts of the input to individual slots. During each iteration, slots compete for the representation of input regions using a softmax-based attention mechanism and refine their representations through a recurrent update function. The final slot representations can then be utilized for downstream tasks such as unsupervised object discovery.

Recently, object-centric models have progressed from learning object representations from synthetic images to videos, such as SAVi (Kipf et al. 2022), SAVi++ (Elsayed et al. 2022), and STEVE (Singh, Y.-F. Wu, and Ahn 2022). Furthermore, recent works, such as DINOSAUR (Seitzer et al. 2023) and VideoSaur (Zadaianchuk, Seitzer, and Martius 2024), have made remarkable progress in scaling object-centric learning to real-world scenes.

Slot Attention for Video (SAVi) utilizes and extends the Slot Attention framework to video data, enabling it to learn object representations, i.e. slots, over temporal sequences. A CNN encoder processes each video frame to generate input feature vectors. Based on these visual features, the Slot Attention module refines the slot representations to produce slots for the current frame. Temporal dynam-

ics and inter-slot interactions are modeled using a Transformer encoder, which propagates the slots to the subsequent time steps. The Spatial Broadcast Decoder (Watters et al. 2019) reconstructs the input frame from the slots, and the model is trained by minimizing the pixel-wise reconstruction error.

DINOSAUR is a recent architecture that leverages the Slot Attention module for image-based object-centric video decomposition. While DINOSAUR adopts an encoder-decoder architecture similar to SAVi, it introduces a key distinction: instead of reconstructing the original input frames, the decoder aims to reconstruct feature representations produced by a DINO ViT encoder (a Vision Transformer pre-trained with the self-supervised DINO method), shifting the focus from pixel-level reconstruction to feature-level learning. Furthermore, besides the MLP decoder, they also introduce a Transformer-based decoder.

3.2 Video Prediction

The task of future frame video prediction involves forecasting upcoming video frames based on preceding frames. A variety of methods have been proposed to address this challenge, leveraging 3D convolutions, recurrent neural networks (RNNs), Transformers, and their combinations. Most of these methods apply an Encoder-Predictor-Decoder framework and can be categorized into four primary architectures: (1) RNN-RNN-RNN, such as PredRNN-V2 (Y. Wang et al. 2023); (2) CNN-RNN-CNN, exemplified by PhyDNet (Guen and Thome 2020) and MSPred (Villar-Corrales, Karapetyan, et al. 2022); (3) CNN-ViT-CNN, as used in the Latent Vision Transformer (Rakhimov et al. 2021); and (4) CNN-CNN-CNN, including the student-teacher network by Chiu, Adeli, and Niebles 2020 and SimVP (Gao et al. 2022). While these methods have demonstrated promising results, they typically model temporal changes using image-level features, without explicitly capturing the compositional structure of video frames or the dynamics of individual objects.

3.2.1 Object-Centric Video Prediction

Object-centric video prediction offers a structured approach that shifts the focus from modeling an entire scene to explicitly predicting the dynamics and states of individual components within it. These methods typically involve three main steps: decomposing the seed video frames into object representations, predicting the future states of these objects using a dynamics model, and rendering the future video frames from the predicted object representations.

The Local Frequency Domain Transformer Network (LFDTN) (Farazi, Nogga, and Behnke 2021) employs local phase differences to separate foreground objects from static backgrounds, modeling their motion in the frequency domain. Another method by Ye et al. 2019 predicts video frames by modeling individual entities in the scene, but this approach relies on supervised annotations of entity locations. Objects-Align-Transition (OAT) (Creswell et al. 2021) combines MONet for scene decomposition with a slot-wise transition module using an alignment mechanism. However, it models object interactions and temporal dynamics using separate modules, with the recurrent dynamics module constrained to a single time-step context window. Object-Centric Video Transformer (OCVT) (Y. Wu, Yoon, and Ahn 2021) utilizes SPACE to decompose scenes into object slots and employs a generative transformer to predict their future states. However, this approach depends on manually disentangled object features and requires error-prone Hungarian matching for latent alignment during training.

In contrast with these works, we propose an object-centric video prediction model that is completely unsupervised, jointly models the object interactions and dynamics across time, spans the context window to multiple time steps, and is independent of the underlying object-centric representations.

Recent advancements in unsupervised object-centric video prediction include OCVP (Villar-Corrales, Wahdan, and Behnke 2023), SlotFormer (Z. Wu et al. 2023), and DDLP (Daniel and Tamar 2024). OCVP and SlotFormer models leverage SAVi for learning object representations (slots) from videos and utilize a Transformer-based predictor to model temporal object dynamics and interactions to predict the subsequent slots.

While SlotFormer uses only a vanilla Transformer encoder as a predictor to jointly process all the input slots, OCVP also proposes two other predictor architectures, OCVP-Seq and OCVP-Par, that decouple the modeling of object interactions and dynamics. This is achieved by employing two specialized multi-head self-attention mechanisms, temporal attention and relational attention. In the temporal attention block, each slot is updated by incorporating information from that same object up to the current time step, without modeling interactions between distinct objects. In contrast, relational attention models the multi-object interactions by jointly processing all slots from the same time step. OCVP-Seq and OCVP-Par differ from each other by the way how they combine the temporal and relational attention blocks, sequentially or in parallel respectively.

Operating in an auto-regressive manner, these models append predicted slots from the current timestep as input to the next timestep. By conditioning on multiple frames, they effectively capture spatio-temporal relationships between objects, ensuring consistency in object properties. Their Transformer-based dy-

namics model predicts future object states based on slot representations, capturing entire objects in single vectors for improved consistency and coherence.

Despite promising results, these models are limited to relatively simple datasets and don't support controllable video predictions. On the other hand, in our work we propose a new decomposition model to learn object representations, replacing SAVi, which allows the model to scale to more complex datasets. Furthermore, we incorporate textual descriptions as additional information to our model, which enable controllable video generation starting from a single image.

3.2.2 Text-Conditioned Video Generation

Recent approaches have integrated textual information into video generation tasks, such as MAGE (Hu, Luo, and Chen 2022), Seer (Gu et al. 2024), and TVP (Song et al. 2024), offering an intuitive means for users to specify desired video content. Text descriptions provide appearance and motion cues that guide the generation of future frames.

MAGE (Motion Anchor-based video Generator) is an auto-regressive framework designed for video generation from a single static image and a textual description. It utilizes a VQ-VAE (Russakovsky et al. 2015) encoder-decoder architecture to learn efficient visual token representations. Cross-attention aligns textual and visual embeddings to produce a spatially-aligned motion representation termed Motion Anchor (MA), which is fused with visual tokens via an axial transformer for video generation. While MAGE demonstrates promising results in controllability, its reliance on VQ-VAE with a high down-sampling ratio leads to potential loss of fine-grained object textures and distortions in overlapping object motions. Moreover, the image-level training of VQ-VAE compromises temporal consistency, resulting in inconsistencies across generated video frames. In contrast, by utilizing object-centric representations and modeling their spatio-temporal relations, our model is able to predict more consistent video frames.

Seer is a diffusion-based model for language-guided video prediction. It employs an Inflated 3D U-Net derived from a pre-trained text-to-image 2D latent diffusion model (Rombach et al. 2022), extending it along the temporal axis and integrating temporal attention layers to simultaneously model spatial and temporal dynamics. For the language conditioning module they introduce a novel approach called Frame Sequential Text (FSText) Decomposer, which decomposes global instructions generated by the CLIP text encoder (Radford et al. 2021), into frame-specific sub-instructions. These are aligned with frames using a transformer-based temporal network and injected into the diffusion process via cross-attention layers.

4 Methodology

In this work, we propose TextOCVP, a novel object-centric model for image-to-video generation guided by language instructions. We extend upon previous object-centric approaches for video generation (Villar-Corrales, Wahdan, and Behnke 2023, Z. Wu et al. 2023) by coping with their limitations in scaling to complex datasets and controllability. This is achieved by adapting a new architecture for object-centric learning and developing a novel predictor architecture that integrates textual information, respectively. To the best of our knowledge, we propose the first object-centric approach for image-to-video generation guided by language.

In this chapter, we give a general overview of our proposed model (Section 4.1) and explain in details its architectural components (Section 4.2). Finally, we present the training and inference procedure of our model (Section 4.3), and the implementation details (Section 4.4).

4.1 Overview

Given an initial reference image $\mathbf{X}_1$ and a text description $\mathbf{C}$, our proposed model generates the subsequent T video frames $\widehat{\mathbf{X}}_{2:1+T}$, which maintain a similar appearance and structural composition as the reference image and follow the motion described in the text caption. We implement an object-centric approach to solve this task. Firstly, the input frame is decomposed into a set of D-dimensional object representations (or slots) $\mathbf{S}_1 \in \mathbb{R}^{N \times D}$, where N is the pre-defined number of slots per-image (Section 4.2.1). Then, these slots are inputted to a Transformer-based predictor which jointly models their spatio-temporal relations, and incorporates the textual information as a guidance for predicting the future object slots $\widehat{\mathbf{S}}_{2:1+T}$ autoregressively (Section 4.2.2). Given the generated slots, the model renders the output video frames $\widehat{\mathbf{X}}_{2:1+T}$ (Section 4.2.3). A general overview of the model is shown in Figure 4.1.

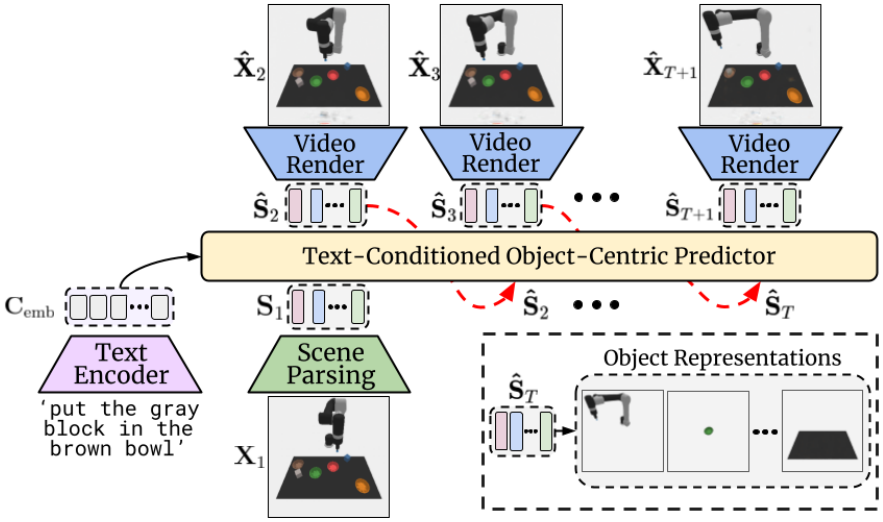


Figure 4.1: General overview of our proposed model for text-driven image-to-video generation.

4.2 Architecture Components

The architecture of our model has two main components: an object-centric model that learns object representations and is leveraged for scene parsing and video rendering, and a text-conditioned predictor. As an object-centric model, we either employ SAVi (Kipf et al. 2022) or a newly proposed encoder-decoder architecture which extends DINOSAUR (Seitzer et al. 2023) to also reconstruct images and work on video data.

4.2.1 Scene Parsing

We employ a scene parsing module to decompose the input image into object representations called slots, where each slot can represent an object on the scene. These slots are permutation-invariant, meaning that the order of the slots is not important and can be changed without affecting the output. Mathematically, given K video frames $\mathbf{X}_{1:K}$, the model decomposes them into object representations $\mathbf{S}_{1:K} = \{S_1, \ldots, S_K\}$, where $\mathbf{S}_t = (\mathbf{s}_t^1, \ldots, \mathbf{s}_t^N) \in \mathbb{R}^{N \times D}$ is the set of D-dimensional object embeddings parsed from frame t. The slots $\mathbf{S}_0$ are initialized by sampling them from a learned Gaussian distribution.

SAVi When using SAVi, at time step t, the input video frame $\mathbf{X}_t$ is encoded via a convolutional neural network (CNN) encoder into multiple feature maps that

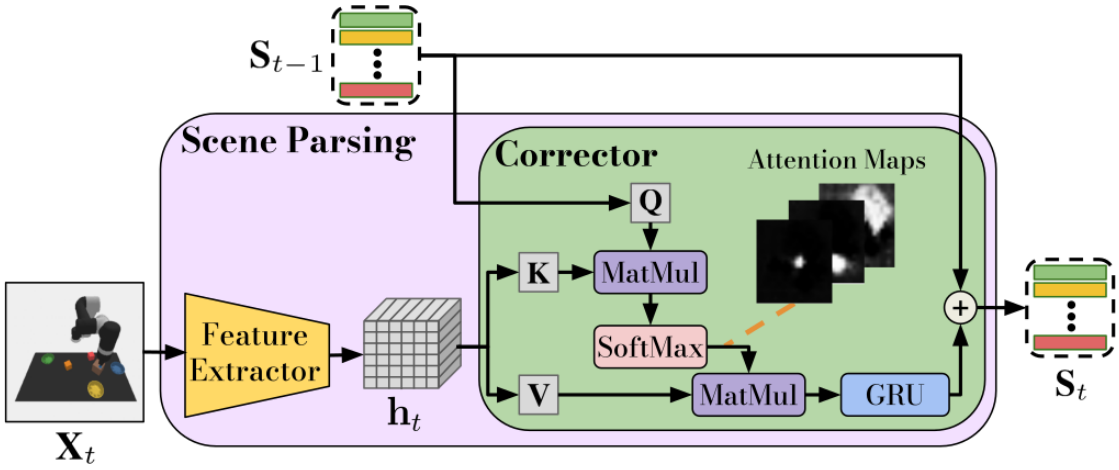


Figure 4.2: Scene parsing procedure, Villar-Corrales, Wahdan, and Behnke 2023

represent semantic high-level information of the input. Positional encodings are added to these feature maps, and then they are spatially flattened and mapped via a shared MLP into feature vectors $\mathbf{h}_t \in \mathbb{R}^{L \times D_{\text{enc}}}$, where L is the number of spatial locations in the flattened feature map, and D_{enc} is the feature dimension.

These feature vectors are fed to a Slot Attention (Locatello et al. 2020) corrector which updates the previous slots $\mathbf{S}_{t-1}$ based on visual features from the current frame $\mathbf{h}_t$, following the iterative attention mechanism. In a single iteration, Slot Attention performs cross-attention, with the attention coefficients softmax-normalized over the slot dimension, thus encouraging the slots to compete to represent parts of the input, and then updates the slot representations using a learned recurrent function, namely Gated Recurrent Unit (GRU) (Cho et al. 2014). This process is visualized in Figure 4.2. The Slot Attention's update step of the previous slots $\mathbf{S}_{t-1}$ is given by:

$$\begin{aligned} A_t &= \underset{N}{\text{softmax}}\left(\frac{q(\mathbf{S}_{t-1})k(\mathbf{h}_t)^T}{\sqrt{D}}\right) \in \mathbb{R}^{N \times L}, \\ U_t &= W_t v(\mathbf{h}_t) \in \mathbb{R}^{N \times D}, W_{i,j} = \frac{A_{i,j}}{\sum_{l=1}^{L} A_{i,l}}, \\ \mathbf{S}_t &= \text{GRU}(U_t, \mathbf{S}_{t-1}), \end{aligned} \tag{4.1}$$

where k, q, v are learned linear projections that map input features and slots to a common dimension D. Layer normalization (LayerNorm) (L. J. Ba, Kiros, and Geoffrey E. Hinton 2016) is applied before each projection. Additionally, for added expressiveness, LayerNorm is applied to the GRU output, which is then passed through an MLP with residual connection. Optionally, this slot refinement step can be iterated multiple times per frame. The final output of this module

is the set of slots $\mathbf{S}_t$, representing the objects of the input frame. The slots are transitioned to be used as input in the next time step by a transition function, implemented as a Transformer encoder, which enables the temporal alignment of slots.

Extended-DINOSAUR DINOSAUR follows a similar procedure to SAVi to parse the input scene into slots. The only difference is that instead of a simple CNN encoder, it employs a Vision Transformer (ViT) (Kolesnikov et al. 2021) pre-trained using a self-supervised algorithm, that we have mainly chosen to be the DINOv2 method (Oquab et al. 2023). Given the input image $\mathbf{X}_t$, ViT extracts patch features $\mathbf{h}_t \in \mathbb{R}^{L \times D_{\text{enc}}}$, where in this case L is the total number of patches, i.e. $L = \frac{H \cdot W}{p^2}$, where H and W are the input image height and width respectively, p is the predefined size of a patch, and D_{enc} is the embedding dimension of a patch feature. Afterwards, each feature is transformed by a small MLP before Slot Attention. In difference with SAVi, positional encodings are not added to the feature vectors before the Slot Attention module, as the ViT's initial positional encodings were found to be sufficient to support spatial grouping. DINOv2 pre-training allows the encoder to extract semantically meaningful features that correlate with the notion of objectness.

4.2.2 Text-Conditioned Predictor

Given the slots $\mathbf{S}_1 \in \mathbb{R}^{N \times D}$ parsed from the reference frame $\mathbf{X}_1$, and the embeddings of a textual description $\mathbf{C}$, the predictor generates the next T set of slots $\widehat{\mathbf{S}}_{2:1+T}$ autoregressively, i.e. the generated slots at time step t are appended to the input in the next time step $t+1$. We propose a Transformer-based architecture as our predictor, which is composed of N_P identical layers, each consisting of three main building blocks, a multi-head self-attention mechanism, a multi-head text-to-slot cross-attention block, and a fully-connected feed-forward network (or MLP). Each building block is preceded by a layer normalization (LayerNorm) and employs residual connections. An overview of the predictor architecture is given in Figure 4.3.

To condition the prediction process, the text description $\mathbf{C}$ is encoded into text token embeddings $\mathbf{C}_{\text{emb}} \in \mathbb{R}^{N_{\text{token}} \times D_{\text{token}}}$, where N_{token} is the number of tokens and D_{token} is the token embedding dimension. We experiment with different variants of the text encoder architecture, including employing a pre-trained T5 (Raffel et al. 2020) encoder or a custom Transformer encoder trained from scratch with the predictor. For each slot sequence generation, the text embeddings are created only once in the beginning.

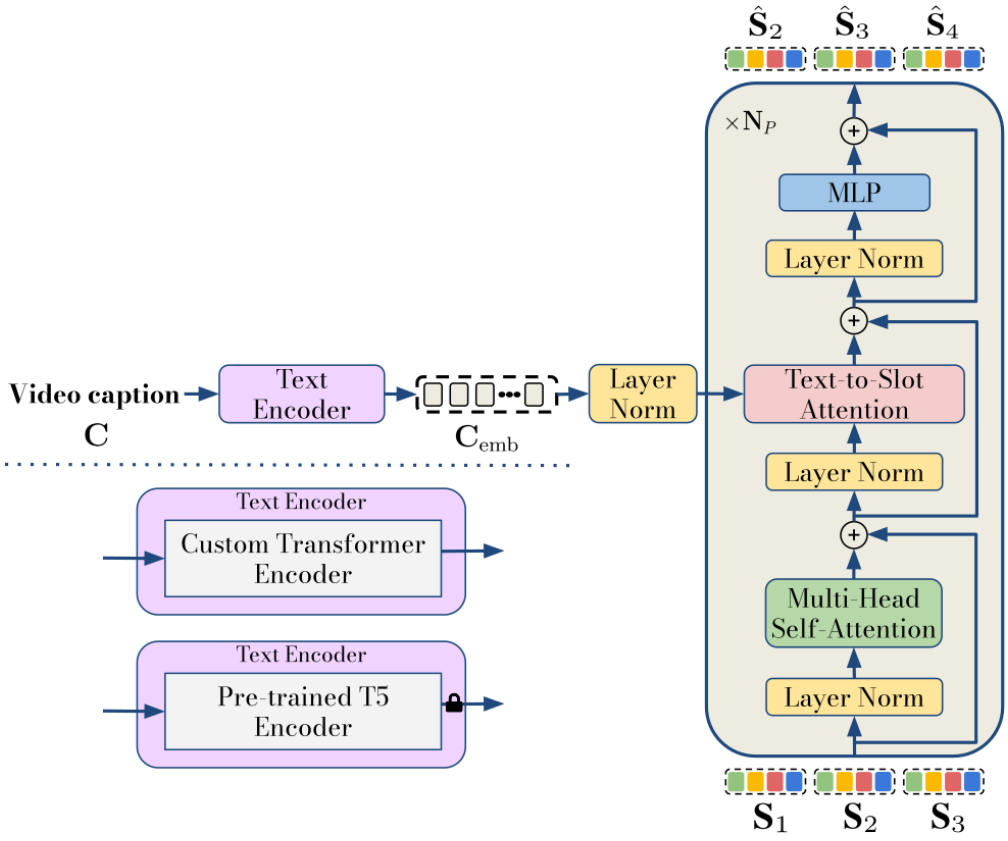


Figure 4.3: Architecture of our proposed predictor and text encoder variants.

At time step t, the predictor receives as input t sets of slots, i.e. the slots $\mathbf{S}_1 \in \mathbb{R}^{N \times D}$ that are parsed from the seed frame and $\widehat{\mathbf{S}}_{2:t} \in \mathbb{R}^{(t-1) \times N \times D}$ which are the slots autoregressively generated by the predictor in the previous time steps, as well as the text embeddings $\mathbf{C}_{\text{emb}}$ of the corresponding language instruction. The input slots are initially mapped via an MLP to the token dimension D_{token}, which serves as the inner dimensionality of the predictor, and learnable temporal positional encodings are added, i.e. all the N slots from the same time step share the same positional encoding, preserving the permutation invariance property. The resulting sequence of slots $\mathbf{S} \in \mathbb{R}^{(t \cdot N) \times D_{\text{token}}}$ is inputted to the predictor.

Our text-conditioned object-centric predictor consists of three layers. The self-attention layer enables every slot to attend to the other slots in the sequence, thus modeling the spatio-temporal relations between them. Afterwards, they are passed through the text-to-slot attention layer, which is a cross-attention module that enhances the slots representations by incorporating important information from the text embeddings, such as the actions and motion. Then, the feed-forward network is independently applied at each position. This process is repeated in

every predictor layer. The last N predicted features are mapped via an MLP back to the original slot dimension D, producing output slots $\widehat{\mathbf{S}}_{t+1} \in \mathbb{R}^{N \times D}$. In most of the experiments, we also apply a final residual connection which improves the temporal consistency of the predictions, i.e. the predicted set of slots for time step t is given as $\widehat{\mathbf{S}}_{t+1} \leftarrow \widehat{\mathbf{S}}_{t+1} + \mathbf{S}_t \in \mathbb{R}^{N \times D}$.

4.2.3 Video Rendering

After the predictor has autoregressively generated the subsequent T slots $\widehat{\mathbf{S}}_{2:1+T}$, with $\widehat{\mathbf{S}}_t = (\widehat{\mathbf{s}}_t^1, \ldots, \widehat{\mathbf{s}}_t^N) \in \mathbb{R}^{N \times D}$, the decoder of the object-centric module is used to render the corresponding video frames $\widehat{\mathbf{X}}_{2:1+T}$.

SAVi decoder SAVi utilizes a slot-wise Spatial Broadcast Decoder (Watters et al. 2019) as object-centric decoder. To reconstruct the video frame $\widehat{\mathbf{X}}_t$ from the set of slots $\widehat{\mathbf{S}}_t$, each individual slot $\widehat{\mathbf{s}}_t^n$ is initially broadcasted to match the spatial size of the image, and positional encoding are added. Then, it is given as input to a CNN decoder to render an object image $\mathbf{o}_t^n$ and an alpha mask $\mathbf{m}_t^n$. The masks are normalized across slots via a softmax function. The video frame is then computed as a weighted sum of the corresponding object images and normalized masks, as described below:

$$\mathbf{o}_t^n, \mathbf{m}_t^n = f_{\text{dec}}(\widehat{\mathbf{s}}_t^n), \qquad \tilde{\mathbf{m}}_t^n = \underset{N}{\text{softmax}}(\mathbf{m}_t^n), \qquad \widehat{\mathbf{X}}_t = \sum_{n=1}^{N} \mathbf{o}_t^n \odot \tilde{\mathbf{m}}_t^n. \quad (4.2)$$

Extended-DINOSAUR decoder When using the newly proposed object-centric module based on the DINOSAUR architecture, we first reconstruct the patch features and then the video frame. To reconstruct the patch features, a Spatial Broadcast Decoder is applied to each slot, similar to SAVi. Given the set of slots $\widehat{\mathbf{S}}_t$, each slot $\widehat{\mathbf{s}}_t^n$ is broadcasted to the number of patches, resulting in L tokens, and learnable positional encodings are added to incorporate the positional information. The resulting tokens are processed by a shared multi-layer MLP, preceded by a layer normalization, that renders the features $\mathbf{y}_t^n \in \mathbf{R}^{L \times D_{\text{enc}}}$ and alpha mask $\mathbf{m}_t^n \in \mathbb{R}^{L \times 1}$. The alpha masks are normalized across slots via softmax to get $\tilde{\mathbf{m}}_t^n$, and the final reconstructed features $\widehat{\mathbf{Y}}_t \in \mathbb{R}^{L \times D_{\text{enc}}}$ are computed as a weighted sum:

$$\widehat{\mathbf{Y}}_\mathbf{t} = \sum_{n=1}^{N} \mathbf{y}_t^n \odot \tilde{\mathbf{m}}_t^n. \quad (4.3)$$

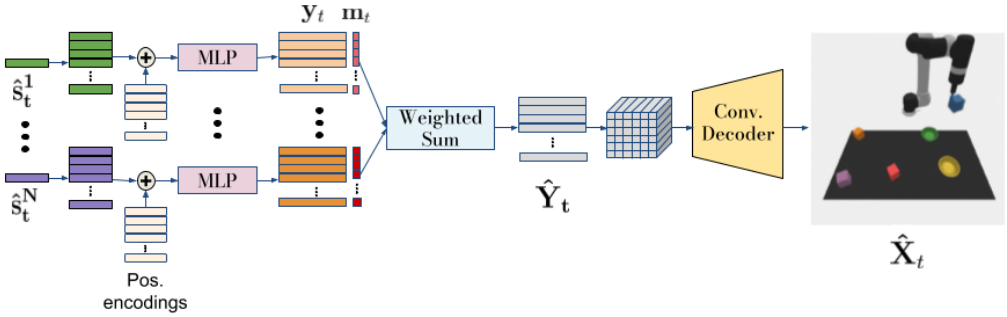


Figure 4.4: The image rendering procedure in the new proposed decoder.

Given the reconstructed patch features, they are organized in a grid shape and then fed to a CNN decoder in order to render the predicted frame $\widehat{\mathbf{X}}_t$. We employ bilinear interpolation to map the image to its pre-defined size, if needed. An overview of the image rendering procedure using our proposed object-centric model's decoder is given in Figure 4.4.

4.3 Training Procedure and Inference

Following common approaches (Villar-Corrales, Wahdan, and Behnke 2023, Z. Wu et al. 2023), we employ a 2-step training procedure for our model, by first training the object-centric learning model, and then the predictor. If the text encoder is not frozen, it is trained/fine-tuned on the predictor training stage.

4.3.1 Object-Centric Learning Training

The goal of the object-centric models is to learn object representations (slots) from video sequences. Both our variants follow an encoder-decoder architecture, but are trained using different reconstruction objectives.

SAVi Given $\mathbf{X}_{1:K}$ input video frames, SAVi processes each frame at a time, by first applying scene parsing (encoder + Slot Attention) to generate the set of

slots, and then it applies the decoder to slots to reconstruct the input image. In this way it outputs the reconstructions $\widehat{\mathbf{X}}_{1:K}$. SAVi is trained by minimizing the mean-squared image reconstruction loss:

$$\mathcal{L}_{\text{SAVi}} = \frac{1}{K}\sum_{k=1}^{K} ||\widehat{\mathbf{X}}_k - \mathbf{X}_k||_2^2. \tag{4.4}$$

Extended-DINOSAUR Seitzer et al. 2023 argue that the image reconstruction objective on its own does not provide sufficient inductive bias to give rise to object groupings when objects have complex appearance and shape. This limits their scalability to real-world datasets. Thus, DINOSAUR tasks its decoder to reconstruct from the slots the self-supervised DINO ViT features produced by its encoder. These features contain a higher-level of semantic information than pixel-level features, and the slots are forced to efficiently encode this information as well.

Inspired by these results, we train our DINOSAUR-based object-centric model by minimizing both the image and feature reconstruction losses. That is, given the input video frames $\mathbf{X}_{1:K}$, the model extracts the patch features $\mathbf{h}_{1:K}$, and the Slot Attention module generates the set of slots $\mathbf{S}_{1:K}$. Given these slots, our proposed decoder reconstructs both the features $\widehat{\mathbf{Y}}_{1:K}$ and the input frames $\widehat{\mathbf{X}}_{1:K}$. The loss is then computed as:

$$\mathcal{L}_{\text{rec}} = \frac{1}{K}\sum_{k=1}^{K} ||\widehat{\mathbf{X}}_k - \mathbf{X}_k||_2^2 + \frac{1}{K}\sum_{k=1}^{K} ||\widehat{\mathbf{Y}}_k - \mathbf{h}_k||_2^2. \tag{4.5}$$

4.3.2 Predictor Training

Given the pre-trained object-centric module, we leverage its encoder and decoder for scene parsing and video rendering for object-centric prediction. Specifically, given the reference frame $\mathbf{X}_1$, it is parsed into object representations $\mathbf{S}_1$ by the object-centric model. The slots are then fed into the predictor, which also incorporates the language instruction that is encoded into text embeddings by the text encoder. The predictor forecasts the slots for the next time step $\widehat{\mathbf{S}}_2$. This process is repeated autoregressively, i.e. the predicted slots are appended to the input in the next time step, to generate the set of slots for all T time steps. This is similar to the autoregressive generation in inference time, and simulating the error accumulation during training leads to better modeling of long-term dynamics in inference (Z. Wu et al. 2023). Therefore, generating slots autoregressively is preferred over using only ground-truth slots as inputs during training.

The predicted set of slots $\widehat{\mathbf{S}}_{2:1+T}$ are rendered by the pre-trained object-centric model's decoder into the generated video frames $\widehat{\mathbf{X}}_{2:1+T}$. The predictor is trained to minimize both the image and slot reconstruction loss:

$$\begin{aligned} \mathcal{L} &= \mathcal{L}_I + \mathcal{L}_S, \\ \mathcal{L}_I &= \frac{1}{T}\sum_{t=1}^{T} ||\widehat{\mathbf{X}}_{1+t} - \mathbf{X}_{1+t}||_2^2, \\ \mathcal{L}_S &= \frac{1}{T \cdot N}\sum_{t=1}^{T}\sum_{n=1}^{N} ||\widehat{\mathbf{s}}_{1+t}^n - \mathbf{s}_{1+t}^n||_2^2, \end{aligned} \tag{4.6}$$

where $\mathcal{L}_I$ is the image reconstruction loss, and $\mathcal{L}_S$ is the slot reconstruction loss. The losses enforce an improvement in the visual quality of generated frames and temporal consistency.

4.3.3 Inference

In inference time, our proposed model takes as input a single reference frame $\mathbf{X}_1$ and the language instruction $\mathbf{C}$. The seed frame is parsed into slots by the object-centric module. The predictor takes as input the parsed slots, and by leveraging the language instruction as guidance, it autoregressively predicts the future set of slots until the pre-defined number of predictions T is reached. Then, the object-centric module is used again to render the generated video frames. The process is illustrated in Figure 4.1.

4.4 Implementation Details

TextOCVP Our main proposed model, TextOCVP, consists of the text-conditioned predictor and the extended version of DINOSAUR as the object-centric module. It also leverages the pre-trained T5 encoder to create the text token embeddings, which is frozen during training.

The predictor is composed of $N_P = 8$ identical layers, each containing a 4-head self-attention mechanism, a 8-head cross-attention module, and an MLP with a single hidden layer of dimension 1024, and a ReLU activation function, which will be the norm for every used MLP (unless stated otherwise). Furthermore, the predictor uses an embedding dimensionality of $d_{\text{model}} = 512$ and applies input context window size of 10. It also employs a final residual connection, which will be used in every model variant.

We leverage the small version of T5 encoder (Raffel et al. 2020), which consists of 6 T5 blocks, where each block employs a self-attention mechanism and feed-forward network, and applies layer normalization and dropout after both sub-blocks. Furthermore, it uses a vocabulary with size 32128.

The Extended-DINOSAUR learns $N = 10$ slots of $D = 128$ dimension. We use DINOv2 ViT-Base as the encoder. It consists of 12 layers, uses patch size $p = 14$, and outputs patch features with dimension $D_{\text{enc}} = 768$. To map the features to the slot dimension, we apply an MLP with a single hidden layer. The Slot Attention module iterates the slot refinement step 3 times for the first frame of the sequence, and then only a single time. We use a single Transformer encoder block as the transition function, which consists of a 4-head self-attention sub-layer, an MLP with hidden dimension 512, and layer normalization applied after the sub-layers. We implement an MLP decoder to reconstruct the patch features from the slots. It has 4 layers and a hidden dimension of 1024. To reconstruct the images from the features, we implement a CNN decoder with 4 layers, where each layer uses 3×3 kernels with stride = 1 and padding = 1. Every layer is followed by an up-sampling function with factor 2, and a final convolutional layer is applied to map to the RGB channels of the image. We train our extended version of DINOSAUR with videos of length $K = 5$ for 1000 epochs, batch size of 16, and learning rate of $4e^{-4}$.

TextOCVP$_{\text{SAVi}}$ We have also implemented another version of our model, named TextOCVP$_{\text{SAVi}}$, that still uses the text-conditioned predictor, but leverages SAVi as the object-centric model. It also uses the same T5 encoder as the main model, with the only difference that it is fine-tuned during the predictor training step.

The predictor used in this variant is very similar to the one used on TextOCVP. They differ only from the hidden dimension of MLP which is 2048 in this case, and the fact that TextOCVP$_{\text{SAVi}}$ uses 8-head self-attention mechanisms.

For every scene, SAVi learns $N = 8$ slots of $D = 128$ dimension. As an encoder for SAVi, we use a 4-layer Convolutional Neural Network (CNN) with 32 kernels in every layer and ReLU activation function, where each kernel has 5×5 spatial dimensions, and uses stride = 1 and padding = 2, i.e. the encoder does not down-sample its input. The Slot Attention module iterates the slot refinement step 3 times only for the first video frame. The transition function is a single-layer Transformer encoder with a 4-head self-attention mechanism, MLP with hidden dimension of 256, and layer normalization applied after each block. For the decoder, we utilize a CNN decoder with 4 convolutional layers with 32 kernels of size 5×5 with stride = 1 and padding = 2, and ReLU activation function, and a final convolutional layer which maps to 4 channels (RGB + alpha mask). SAVi

Table 4.1: An overview of the predictor architecture and text encoder employed in every variant of our proposed model.

Variant	# Layers	Emb. Dim.	# Heads SA	# Heads CA	Residual	MLP Dim.	Context	Text Encoder
TextOCVP	8	512	4	8	Yes	1024	10	T5 - frozen
TextOCVP$_{\text{SAVi}}$	8	512	8	8	Yes	2048	10	T5 - FT
TextOCVP$_{\text{SAVi}_S}$	2	512	4	4	Yes	256	5	T5 - FT
TextOCVP$_{\text{SAVi}_C}$	2	128	4	4	Yes	256	5	Custom

is trained using video sequences of $K = 10$ frames for 1000 epochs, using batch size of 64, and learning rate of $1e^{-4}$.

TextOCVP$_{\text{SAVi}}$ variants We also employ two smaller variants of TextOCVP$_{\text{SAVi}}$, which use the same SAVi model as described above, but differ from the text-conditioned predictor and/or text encoder that they use.

TextOCVP$_{\text{SAVi}_S}$ ("S" refers to "small") uses the same text encoder, but employs a small predictor which consists of only $N_P = 2$ layers. Each layer contains a 4-head self-attention mechanism, a 4-head cross-attention block, and an MLP with hidden dimension of 256. It has an inner embedding dimension of $d_{\text{model}} = 512$, and context window size of 5.

TextOCVP$_{\text{SAVi}_C}$ ("C" refers to "custom") leverages a custom Transformer-based text encoder, which is trained from scratch in the predictor training step. It consists of 2 layers which are composed of a 4-head self-attention mechanism and a feed-forward network with hidden dimension of 512 and GELU activation function (Hendrycks and Gimpel 2016). Every sub-layer is followed by a layer normalization and dropout is applied. The custom text encoder employs a vocabulary of size 30. SAVi used in TextOCVP$_{\text{SAVi}_C}$ is trained to learn $N = 6$ slots. Furthermore, this variant follows the same predictor architecture as TextOCVP$_{\text{SAVi}_S}$, with the only difference that it uses an inner dimension of $d_{\text{model}} = 128$. An overview of the predictor architecture and text encoder used in every variant is given in Table 4.1.

Training details For every training procedure, for both the object-centric module and the predictor, we use Adam optimizer (Kingma and J. Ba 2015), learning rate warmup, cosine annealing as a learning rate scheduler, and gradient clipping. All our models are implemented in PyTorch and trained on a single NVIDIA A6000 (48Gb).

5 Experiments

In this chapter, we illustrate our experimental results. We begin by first describing the setup where we evaluate the experiments, i.e. the datasets and metrics (Section 5.1). Then we present quantitative and qualitative comparisons of our TextOCVP model and the baselines for various tasks and datasets (Section 5.2). Finally, we conduct an in-depth analyze of our model, including ablation studies about design choices and demonstrations of controllability and model robustness (Section 5.3).

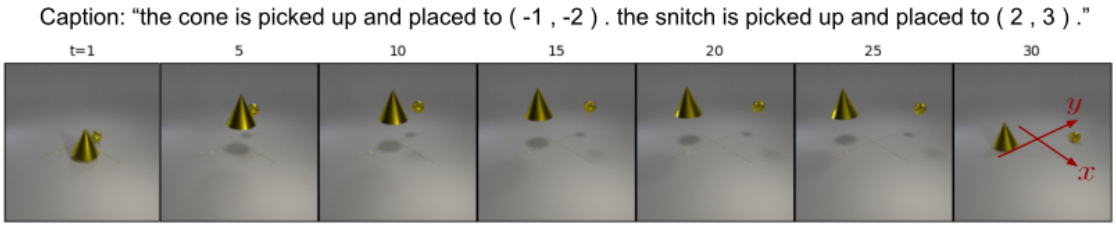
5.1 Experimental Setup

In this section, we will explain in detail the experimental setup, including the datasets and metrics used to evaluate the performance of our model and other approaches for video generation.

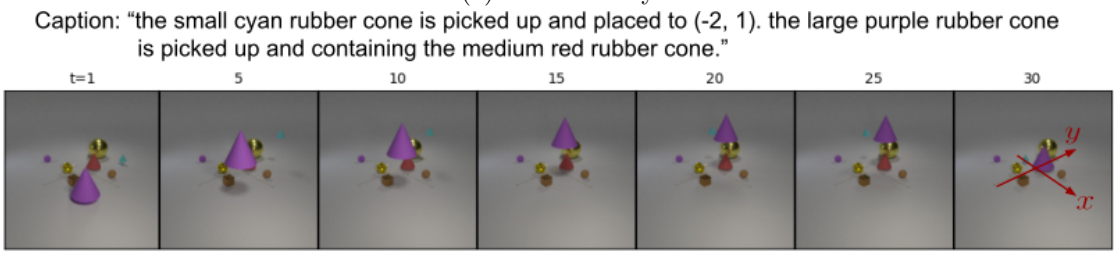
5.1.1 Datasets

We evaluate the performance of TextOCVP for text-driven image-to-video generation on CATER (Girdhar and Ramanan 2020) and CLIPort (Shridhar, Manuelli, and Fox 2021) datasets. On CATER, we also evaluate our model for the video prediction task, i.e., leveraging multiple seed video frames to predict the future video sequence. A description of both datasets is given below.

CATER CATER is a dataset that consists of long video sequences, each described by a textual caption. The video scenes consist of multiple 3D geometric objects in a 2D table plane, which is split into a 6×6 grid with fixed axis, allowing the exact description of object's positions using coordinates. The text instruction describes the movement of specific objects through four atomic actions: 'rotate', 'pick-place', 'slide', and 'contain'. The caption follows a template consisting of the subject, action, and an optional object or end-point coordinate, depending on the action. For example, the final coordinates are included for 'pick-place' and 'slide' actions. The movement of the objects starts at the same time step. Furthermore, the initial position of objects is randomly selected from the plane



(a) CATER-easy



(b) CATER-hard

Figure 5.1: Examples of video sequences with their corresponding caption in (a) CATER-easy and (b) CATER-hard.

grid, and the camera position is fixed for every sequence. Following Hu, Luo, and Chen 2022, we use two variants of this dataset, namely CATER-easy and CATER-hard, illustrated in Figure 5.1. For both variants, we set the spatial size of the video frames to 64×64.

CATER-easy is a simple version of the CATER dataset, consisting of 3500 video-caption pairs for training and 1500 test pairs. Every video consists of only two objects, a cone and a snitch, which is a special small object in metallic gold color, shaped like three intertwined toruses. All four actions can be applied, with the constrain that the 'rotate' action can be afforded only by the snitch, while the 'contain' action only by the cone. Every video can include the movement of only one of the objects, or the simultaneous movement of both. The vocabulary size is 30.

CATER-hard is a more complete and larger version of the CATER dataset, containing 24000 video-caption training pairs and 6000 testing pairs. Besides the cone and the snitch, it also includes the cube, sphere, and cylinder objects. Furthermore, every object is described by its size (small, medium, or large), material (metal or rubber), and color (red, blue, green, yellow, gray, brown, purple, cyan, or gold if the object is the snitch), and this description is included in the textual caption. As in CATER-easy, every atomic action is available. The 'rotate' action is afforded by cubes, cylinders and the snitch, the 'contain' action is again only afforded by the cones, while the other two actions are afforded by every object.

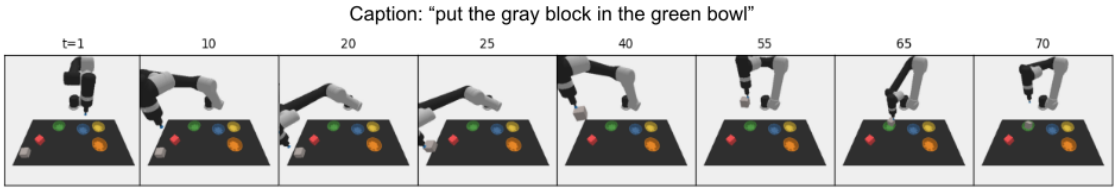


Figure 5.2: Example of a video-caption pair in the CLIPort dataset.

Every video has between 3 and 8 objects, and two actions happen to different objects at the same time. The vocabulary size is 50.

CLIPort CLIPort is a robot manipulation dataset, consisting of video-caption pairs, i.e. long videos whose motion is described by a textual video caption. There are many variants of the CLIPort dataset, but we focus on the *Put-Block-In-Bowl* variant. Following Shridhar, Manuelli, and Fox 2021, we generate 20000 training pairs, and 500 validation and testing pairs, with resolution 320×360, which we later interpolate to 336×336.

Every video contains 6 objects on a 2D table plane, and a robot arm. Objects can be either a block or a bowl, and there is at least one of them in every sequence. The starting position of each object is random, with the only constrain to be placed on the table. Each video describes the action of the robot arm picking a block, and putting it in a specific bowl, as illustrated in Figure 5.2. The video caption follows the template "put the [color] block in the [color] bowl". Each individual object in the scene has a different color. In the train and validation set, the block and the bowl that are part of the caption can have one of the following colors: blue, green, red, brown, cyan, gray, or yellow, while in the test set they can have blue, green, red, pink, purple, white, or orange color. The other 4 objects, called distractors, can have any color. During a video sequence, it can be possible that the robot arm goes out of frame, and comes back in later frames, thus requiring the model to leverage long range dependencies. The vocabulary size is 15.

5.1.2 Metrics

To measure the performance of our proposed model, and compare it with other approaches, we evaluate the visual quality of predicted video frames. This is achieved through perceptual metrics, such as Peak Signal-to-Noise Ratio (PSNR), Structural SIMilarity (SSIM) (Z. Wang et al. 2004), and Learned Perceptual Image Patch Similarity (LPIPS) (Zhang et al. 2018). These metrics are used to assess the perceptual similarity between a ground-truth image $\mathbf{X}$, and a predicted frame $\widehat{\mathbf{X}}$.

PSNR PSNR is a classic metric that evaluates the similarity of two images by comparing their pixel values to measure the quality degradation. It is very simple to calculate and has a clear physical meaning, features that contribute to its wide usage. PSNR is calculated as the ratio of peak signal power L to the mean squared error:

$$\begin{aligned} \mathrm{MSE}(\mathbf{X}, \widehat{\mathbf{X}}) &= \frac{1}{H \cdot W} \sum_{i,j} ||\mathbf{X}_{ij} - \widehat{\mathbf{X}}_{ij}||_2^2, \\ \mathrm{PSNR}(\mathbf{X}, \widehat{\mathbf{X}}) &= 10 \log_{10} \left(\frac{L^2}{\mathrm{MSE}(\mathbf{X}, \widehat{\mathbf{X}})} \right), \end{aligned} \tag{5.1}$$

where H and W are the image height and width respectively, and we set $L = 1$. Despite its popularity, PSNR does not align well with the perceived visual quality. One common example is blurring, which causes a small change in the difference between pixel values, but a large perceptual change, which PSNR is not able to represent. Furthermore, it assumes pixel-wise independence, making it insufficient to evaluate structured entities like images.

SSIM SSIM is a perceptual metric that measures the structural similarity between the original and the degraded image. More specifically, it performs comparisons about luminance, contrast, and structure, which are combined to a single equation that represents the structural similarity:

$$\mathrm{SSIM}(\mathbf{X}, \widehat{\mathbf{X}}) = \frac{(2\mu_{\mathbf{X}}\mu_{\widehat{\mathbf{X}}} + C_1)(2\sigma_{\mathbf{X}\widehat{\mathbf{X}}} + C_2)}{(\mu_{\mathbf{X}}^2 + \mu_{\widehat{\mathbf{X}}}^2 + C_1)(\sigma_{\mathbf{X}}^2 + \sigma_{\widehat{\mathbf{X}}}^2 + C_2)}, \tag{5.2}$$

where $\mu_{\mathbf{X}}$, $\mu_{\widehat{\mathbf{X}}}$, $\sigma_{\mathbf{X}}^2$, $\sigma_{\widehat{\mathbf{X}}}^2$, and $\sigma_{\mathbf{X}\widehat{\mathbf{X}}}$, are computed as a smoothing convolution over $\mathbf{X}$, $\widehat{\mathbf{X}}$, $(\mathbf{X} - \mu_{\mathbf{X}})^2$, $(\widehat{\mathbf{X}} - \mu_{\widehat{\mathbf{X}}})^2$, and $(\mathbf{X} - \mu_{\mathbf{X}})(\widehat{\mathbf{X}} - \mu_{\widehat{\mathbf{X}}})$, respectively. In this way, SSIM calculates local measures of similarity by comparing small image patches and then computes the average similarity across the entire image. SSIM is often regarded as a more suitable metric to represent perceived visual quality compared to PSNR, as it aligns better with human visual perception. However, it does not adapt well to scenarios where spatial ambiguities are present (Sampat et al. 2009).

LPIPS In contrast with the other two metrics that compare images by their pixel values in the image space, LPIPS operates on the image features in a latent space. Specifically, it computes the perceptual similarity of two images by calculating the $\mathcal{L}_2$ norm between their features, extracted using a deep convolutional neural network, pre-trained on the image classification task. These features, by being learned to solve a semantic prediction task, are representations in which the Eu-

clidean distance is highly predictive of perceptual similarities (Zhang et al. 2018). The LPIPS metric between two images can be computed as:

$$\text{LPIPS}(\mathbf{X}, \widehat{\mathbf{X}}) = \sum_{l \in \mathcal{F}} \frac{1}{H_l \cdot W_l} \sum_{i,j} ||w_l \odot (\tilde{\mathbf{X}}^l_{ij} - \tilde{\widehat{X}}^l_{ij})||_2^2, \tag{5.3}$$

where $\tilde{\mathbf{X}}^l_{ij}$ and $\tilde{\widehat{X}}^l_{ij}$ are the normalized feature vectors outputted from an intermediate layer l in the perceptual network $\mathcal{F}$, i.e. AlexNet (Krizhevsky 2014), and w_l are the learned weights for layer l. It is shown that LPIPS computes perceptual similarities between two images that align better human perception compared to PSNR and SSIM (Sara, Akter, and Uddin 2019), by successfully capturing the underlying semantics of images.

5.2 Evaluation

In this section we describe our experimental results on both CATER and CLIPort datasets. We conduct a quantitative and qualitative evaluation of our model by also comparing it with the respective baselines.

5.2.1 Experiments on CATER

Our experiments on both variants of the CATER dataset, CATER-easy and CATER-hard, focus on emphasizing the importance of text information in enhancing the prediction of future slots and allowing control over these predictions. Following this motivation, we utilize $\text{TextOCVP}_{\text{SAVi}}$, and its variants, for experiments on CATER dataset, i.e. we use our newly proposed text-conditioned predictor and, similar to other object-centric approaches (Villar-Corrales, Wahdan, and Behnke 2023, Z. Wu et al. 2023), we employ SAVi to learn object representations.

We evaluate the performance of our model in the video prediction task, following a similar setup as other object-centric approaches, and in the text-driven image-to-video generation task, which is the main focus of our work, and also allows us to compare with other text-conditioned approaches.

Implementation Details

Due to the simplicity of dynamics in its sequences, for experiments on CATER-easy, we leverage $\text{TextOCVP}_{\text{SAVi}_C}$ variant, which consists of a two-layer predictor and a custom text encoder. On CATER-hard experiments, we employ $\text{TextOCVP}_{\text{SAVi}_S}$ for video prediction task and $\text{TextOCVP}_{\text{SAVi}}$ for image-to-video generation task.

All the variants are trained for 1500 epochs in their respective datasets, using batch size of 64 and learning rate of $1e^{-4}$.

As baselines, for the video prediction task, we use two variants of object-centric video predictors from Villar-Corrales, Wahdan, and Behnke 2023, namely OCVP-Transformer and OCVP-Seq, trained using the same SAVi as our models. OCVP-Transformer uses a vanilla Transformer encoder as predictor, which has 2 layers with 4-head self-attention modules and MLP with hidden dimension of 256. It employs a final residual connection and context window size of 5. OCVP-Seq uses a 4-layer predictor with two specialized 4-head self-attention mechanisms (for more details refer to Section 3.2.1). It also employs a final residual connection and context window size of 10.

For the image-to-video generation task, besides OCVP-Seq which is not conditioned on textual information, we also explore various text-conditioned approaches as baselines. We employ a non-object-centric baseline (TextOCVP w/o OC), which we implement similarly to our $\text{TextOCVP}_{\text{SAVi}}$ model, with the difference that its SAVi module learns only a single slot of dimension $D = 512$ and utilizes a ResNet-34 as encoder. It uses the same predictor architecture as $\text{TextOCVP}_{\text{SAVi}}$, and uses a T5 text encoder which is freezed during training.

We also employ MAGE (Hu, Luo, and Chen 2022) and SEER (Gu et al. 2024) as baselines for the text-driven image-to-video generation task. We trained these models on CATER dataset and tried to follow rigorously their original implementation. For MAGE, we utilize a 512×256 codebook and a down-sampling ratio of 4. Furthermore, to keep the setting as similar as ours, we discard the speed parameter, i.e. we train the model with subsequent video frames. We initialize SEER from a checkpoint pre-trained on the Something-Something V2 dataset (Goyal et al. 2017), and further fine-tune for 16 epochs on CATER. We also found that using the text loss improved the model performance, while other parameters are similar to the original implementation.

Video Prediction

Video prediction task differs from image-to-video generation as it takes multiple video frames as input to predict the future frames. Formally, given C seed frames, the model has to predict the future T video frames. We train our predictor and the baselines using $C = 5$ seed frames to predict the next $T = 5$ video frames.

Table 5.1 shows the quantitative evaluation on CATER datasets for the video prediction task, demonstrating the visual quality for $T = 15$ predicted video frames during inference. We can observe that in both CATER variants, our proposed model outperforms the object-centric baselines in every metric. Since all the mod-

els use the same pre-trained SAVi as an object-centric model, these results indicate the importance of textual information in guiding deterministic predictions closer to the ground truth.

Table 5.1: Quantitative evaluation on CATER-easy and CATER-hard for video prediction task.

Method	**CATER-easy**$_{5\rightarrow15}$			**CATER-hard**$_{5\rightarrow15}$		
	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
OCVP-Transformer	30.644	0.906	0.071	30.048	0.881	0.060
OCVP-Seq	<u>31.637</u>	<u>0.915</u>	<u>0.061</u>	<u>30.856</u>	<u>0.894</u>	<u>0.054</u>
TextOCVP$_{\text{SAVi}}$	**33.557**	**0.938**	**0.034**	**33.363**	**0.930**	**0.031**

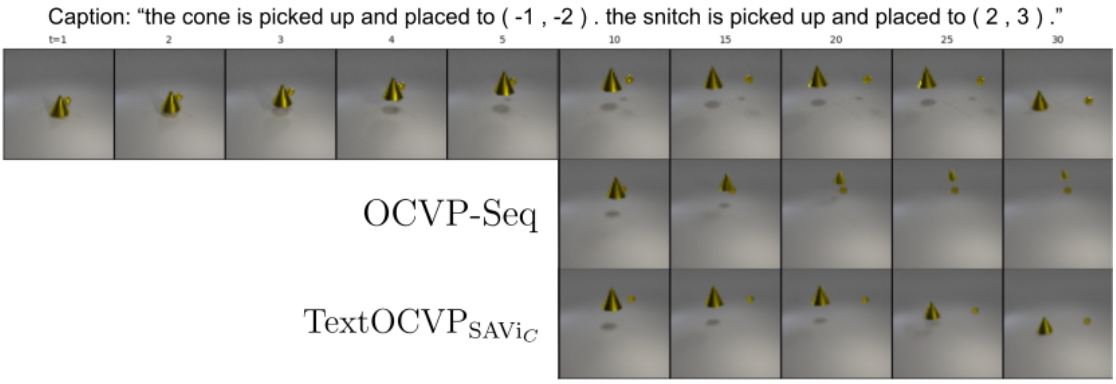
This behavior is demonstrated during qualitative evaluation of the models, as shown in Figure 5.3. Figure 5.3a shows that despite the simple sequence, OCVP-Seq is unable to correctly predict the future behavior of neither the cone nor snitch. This is further emphasized on CATER-hard (Figure 5.3b), where again our model, by leveraging the video caption as an additional information, is able to correctly predict the future motion of the objects.

Image-to-Video Generation

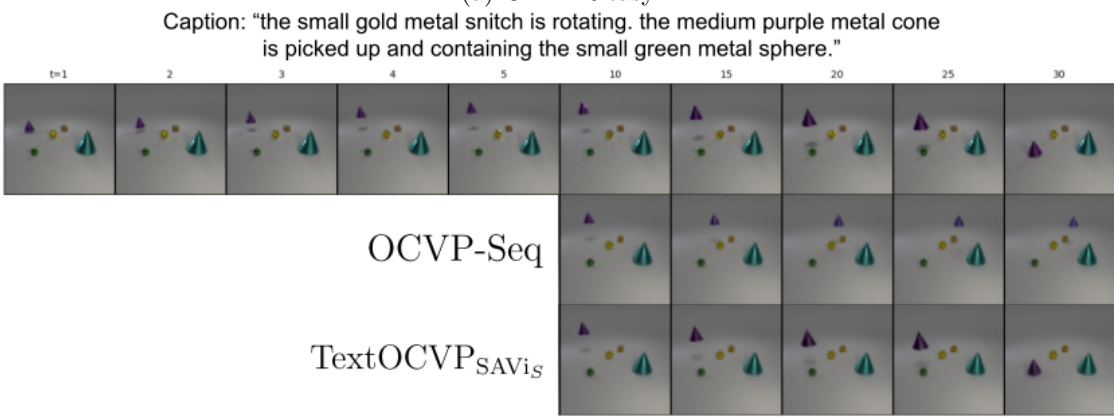
The image-to-video generation task aims to generate a video sequence, given only a single reference frame and a text description. We train the models to generate $T = 9$ frames, and scale this number during inference to observe how the models adapt.

We conduct quantitative evaluations on CATER-hard using the same setting as in training, i.e. predicting $T = 9$ frames, as well as when predicting $T = 19$ future frames, as shown in Table 5.2. In both settings, our TextOCVP$_{\text{SAVi}}$ model outperforms all other models on SSIM and LPIPS metrics. Our model demonstrates significantly improved perceptual quality, with an LPIPS score of 0.035, which is more than two times better than the other models. Meanwhile, MAGE performs best on PSNR metric, which, as stated earlier, does not align well with the human visual perception.

Qualitative evaluations on our model and MAGE illustrate this behavior. Figure 5.4 shows the predicted sequences starting from a single image and guided by the video caption. As can be observed, the predicted frames of our model are closely aligned to the ground-truth, while MAGE predictions introduce many artifacts, such as: missing objects, blurry contours, and significant changes on object shapes.



(a) CATER-easy



(b) CATER-hard

Figure 5.3: Qualitative evaluation for video prediction task on (a) CATER-easy and (b) CATER-hard.

In Figure 5.5, we show the motion prediction in a simple scene, where both MAGE and TextOCVP$_{\text{SAVi}}$ perform well by correctly predicting the motion of the brown cone. However, our model needs more time frames to predict this motion, leading to dissimilarities with the ground-truth pixel values. This behavior can also affect the PSNR score, which is solely based on pixel values differences.

5.2.2 Experiments on CLIPort

We also conduct experiments on the CLIPort dataset, where we employ our main model, TextOCVP. We utilize the newly proposed object-centric module and emphasize its importance in scaling to more complex datasets. We evaluate TextOCVP for image-to-video generation task guided by language, and do a thorough comparison with other relevant non-object-centric approaches, such as MAGE and SEER.

Table 5.2: Quantitative evaluation on CATER-hard for image-to-video generation task.

Method	CATER-hard$_{1\to9}$			CATER-hard$_{1\to19}$		
	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
OCVP-Seq	29.078	0.874	0.078	28.107	0.854	0.101
TextOCVP w/o OC	29.682	0.874	0.092	28.390	0.849	0.112
SEER	22.051	0.723	0.245	16.045	0.535	0.299
MAGE	**34.913**	0.877	0.108	**34.763**	0.871	0.111
TextOCVP$_{\text{SAVi}}$	32.825	**0.924**	**0.035**	31.288	**0.904**	**0.042**

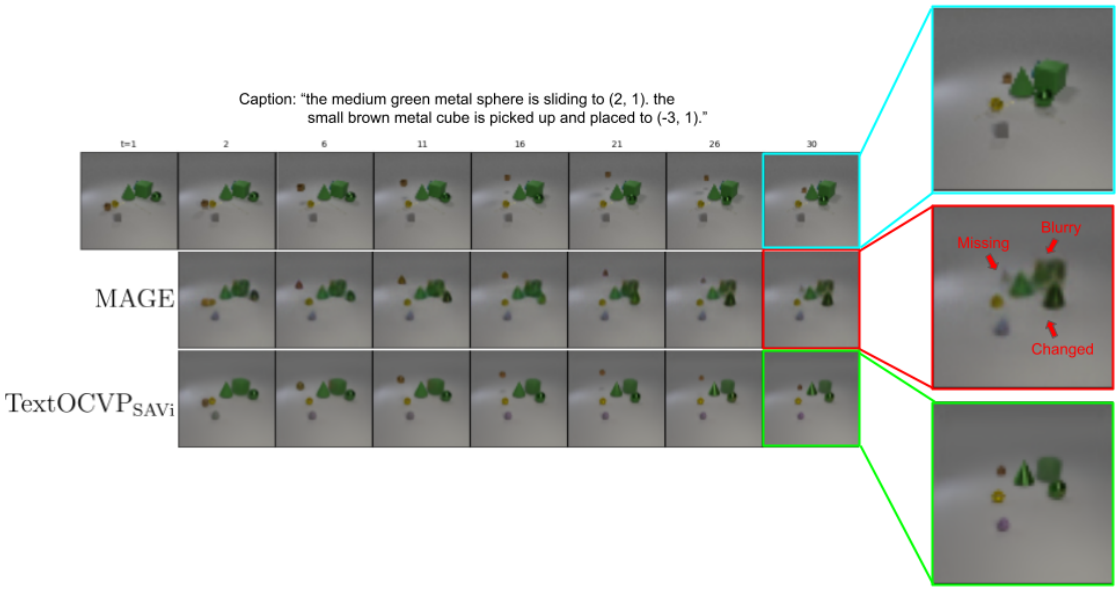


Figure 5.4: Qualitative evaluation for text-driven image-to-video generation on CATER-hard.

Implementation Details

For our model, we employ TextOCVP, which consists of 8-layer text-conditioned predictor and Extended-DINOSAUR as an object-centric model. For the baselines, we implement a non-object-centric model which uses the same predictor, text encoder, and Extended-DINOSAUR architecture as TextOCVP. The only difference is that its object-centric module learns a single slot of dimension $D = 512$, instead of $N = 10$ slots of $D = 128$ dimension. By following this setting, we can compare the importance of representing each object in the scene with a distinct slot. We

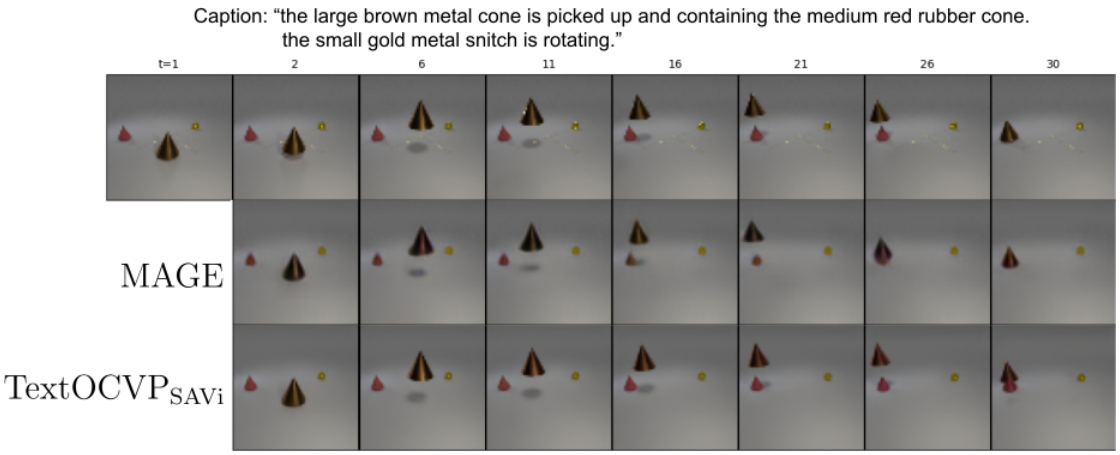


Figure 5.5: An example where our TextOCVP$_{\text{SAVi}}$ model is "slower".

train our model for 700 epochs on CLIPort, with batch size of 10 and learning rate of $4e^{-4}$.

For MAGE, we use a codebook of size 512×1024, and again omit the speed. Furthermore, we replace the standard CNN encoder and decoder used in VQ-VAE with the DINOv2 ViT encoder and CNN decoder used in our TextOCVP approach. This replacement not only allows for a fairer comparison, but was also found crucial in improving the MAGE performance on CLIPort. We named this new version MAGE$_{\text{DINO}}$. We again leverage the pre-trained SEER on the Something-Something V2 dataset, and further fine-tune on CLIPort for 8 epochs, by also considering the text loss. The other parameters were kept the same with the original implementation.

Image-to-Video Generation

All the models are trained to predict $T = 9$ frames during training, provided the reference frame and the video caption. We evaluate the models' performance in the same setting as in training, and in the setting of generating more frames.

First, we argument the need of adapting the original MAGE architecture to MAGE$_{\text{DINO}}$ for video generation on CLIPort. As it is shown in Table 5.3, MAGE completely fails to converge and MAGE$_{\text{DINO}}$ clearly outperforms it.

Table 5.4 shows a detailed quantitative evaluation for image-to-video generation task on CLIPort dataset. We can observe that our TextOCVP model outperforms the other models on every metric when evaluated on the same setting as in training. When generating more future frames, our model shows competitive performance with MAGE$_{\text{DINO}}$, which seems to be more consistent when generating for longer sequences.

Table 5.3: Comparison of $\text{MAGE}_{\text{DINO}}$ with original MAGE implementation for image-to-video generation on CLIPort

Method	$\textbf{CLIPort}_{1\rightarrow 9}$		
	PSNR↑	SSIM↑	LPIPS↓
MAGE	7.116	0.453	0.713
$\textbf{MAGE}_{\textbf{DINO}}$	**23.723**	**0.940**	**0.064**

Table 5.4: Quantitative evaluation on CLIPort for text-driven image-to-video generation task.

Method	$\textbf{CLIPort}_{1\rightarrow 9}$			$\textbf{CLIPort}_{1\rightarrow 19}$		
	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
TextOCVP w/o OC	23.442	0.901	0.184	20.139	0.872	0.210
SEER	21.011	0.887	0.141	11.297	0.622	0.331
$\textbf{MAGE}_{\textbf{DINO}}$	<u>23.723</u>	<u>0.940</u>	<u>0.064</u>	<u>22.265</u>	**0.931**	**0.075**
TextOCVP	**26.995**	**0.950**	**0.062**	**23.884**	**0.931**	<u>0.078</u>

During qualitative evaluations, we observe that this assumption does not appear to hold true in many cases. In Figure 5.6, we illustrate the generation of a full video sequence from TextOCVP and $\text{MAGE}_{\text{DINO}}$, and compare it with the ground-truth sequence which appears in the top row. We can observe in Figure 5.6a that the predictions of $\text{MAGE}_{\text{DINO}}$ initially appear similar to the ground-truth frames, but then the robot arm gets stuck and the gray block is not put on the brown bowl. Meanwhile, our model correctly generates the sequence following the given language instruction. In Figure 5.6b, we can see that TextOCVP predicts the motion in the scene more precisely, while $\text{MAGE}_{\text{DINO}}$ misplaces the blue block on the edge of the bowl and the block is later disappeared during the last frames. Furthermore, the shapes of objects in the scene predicted by our model are more refined, while $\text{MAGE}_{\text{DINO}}$ loses the fine-grained details of the objects, especially blocks, during reconstruction.

5.3 Model Analysis

In this section, we provide various ablation studies which led to the final version of our models described earlier. Furthermore, we conduct an in-depth analysis of the performance of our model, including the control over its predictions and analysis of its interpretability and robustness.

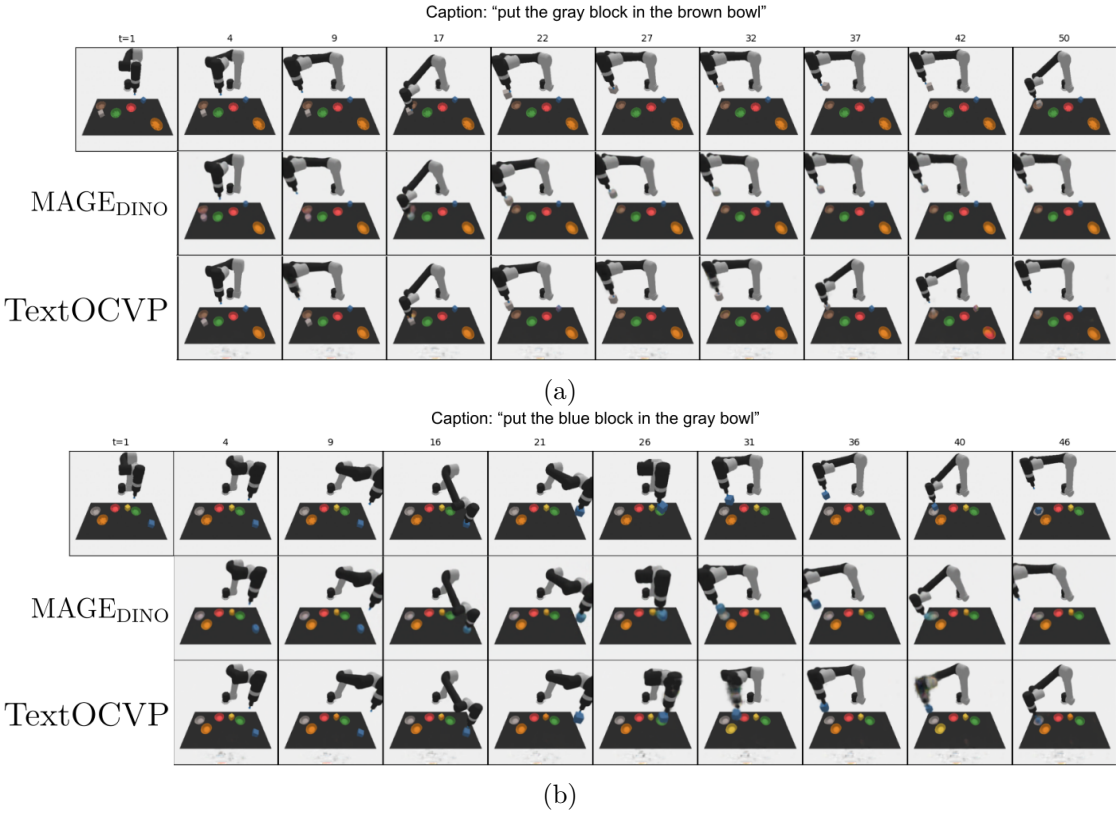


Figure 5.6: Qualitative evaluation for text-driven image-to-video generation task on CLIPort.

5.3.1 Ablation Studies

Predictor Architecture. We experimented with various predictor architectures, which primarily differ from how and where they incorporate the textual information to enhance the slot representations. Specifically, we experimented on applying the cross attention module only once to the slots of the last seed frame, and appending the resulting representation to input slots in every time step ("Naive" approach). Then, we tried incorporating the text information to every slot on the predictor input ("Input" approach), or to the output slots after the predictor ("Output" approach). Table 5.5 provides an overview of the performance of our model on the video prediction task on CATER-easy, using different predictor architectures. As can be observed, the model using the "EveryLayer" approach, which incorporates the cross attention module on every predictor layer, has the best performance.

Number of Layers at Predictor. We studied the impact of number of layers in the predictor. Table 5.6 shows the performance of TextOCVP$_{SAVi}$ on image-

Table 5.5: Comparison of different predictor architectures.

Approach	CATER-easy$_{5\rightarrow15}$		
	PSNR↑	SSIM↑	LPIPS↓
Naive	32.766	0.932	0.038
Input	32.505	0.928	0.041
Output	33.280	0.936	0.035
EveryLayer	**33.557**	**0.938**	**0.034**

to-video generation on CATER-hard, using predictors with different number of layers. We can observe that using the predictor with 8 layers provides the best performance. We believe that further scaling the predictor by increasing the number of layer would barely improve the performance, judging the low improvement gained when scaling from 4 to 8 layers.

Table 5.6: Comparison of model performance using predictors with different number of layers.

Number of Layers	CATER-hard$_{1\rightarrow9}$		
	PSNR↑	SSIM↑	LPIPS↓
2	31.832	0.912	0.041
4	32.580	0.921	0.036
8	**32.825**	**0.924**	**0.035**

Final Residual Connection at Predictor. In every experiment, we apply a final residual connection to the output slots of the predictor. We explored its effect on the TextOCVP model performance on image-to-video generation on CLIPort. As seen in Table 5.7, the performance of the model is improved when applying the residual connection.

Table 5.7: Evaluation of the residual connection's impact on model performance.

Residual	CLIPort$_{1\rightarrow9}$		
	PSNR↑	SSIM↑	LPIPS↓
No	26.556	0.946	0.066
Yes	**26.995**	**0.950**	**0.062**

Text Encoder. We proposed 2 main approaches for the text encoder, a custom Transformer encoder or a pre-trained T5 encoder which can be frozen or fine-tuned during the second stage training. We have leveraged these approaches in different variants of our model, as depicted in Table 4.1. We further investigated the impact of text encoder architecture on performance of the models.

In Table 5.8, we show the generation quality of models that use the same predictor and object-centric model architecture, but different text encoder approaches. We evaluate their performance for the video prediction task on CATER-hard. As can be seen, the model that utilizes the fine-tuned T5 text encoder marginally provides the best performance.

Table 5.8: Evaluation on the impact of text encoder on the model performance on CATER-hard.

Text Encoder	**CATER-hard**$_{5\rightarrow15}$		
	PSNR↑	SSIM↑	LPIPS↓
Custom	32.543	0.920	0.035
Frozen T5	33.250	0.929	**0.031**
Fine-Tuned T5	**33.363**	**0.930**	**0.031**

However, we observed that this conclusion does not always hold. In Table 5.9, we analyze the TextOCVP performance on CLIPort, when using a frozen or fine-tuned T5 text encoder. The model that freezes T5 during training, which is our main model, outperforms its counterpart.

From these observations, we draw the conclusion that the choice of text encoder depends on the task and dataset, especially on its vocabulary size and complexity. Generally, we recommend the initial usage of frozen T5 text encoder, and if the performance is not sufficient, further fine-tune it during predictor training.

Table 5.9: Evaluation on the impact of text encoder on the model performance on CLIPort.

Text Encoder	**CLIPort**$_{1\rightarrow9}$		
	PSNR↑	SSIM↑	LPIPS↓
Frozen T5	**26.995**	**0.950**	**0.062**
Fine-Tuned T5	26.854	0.947	0.065

SAVi vs DINOSAUR. Current object-centric approaches for video generation are limited to relatively simple synthetic datasets. We claim that one of the main

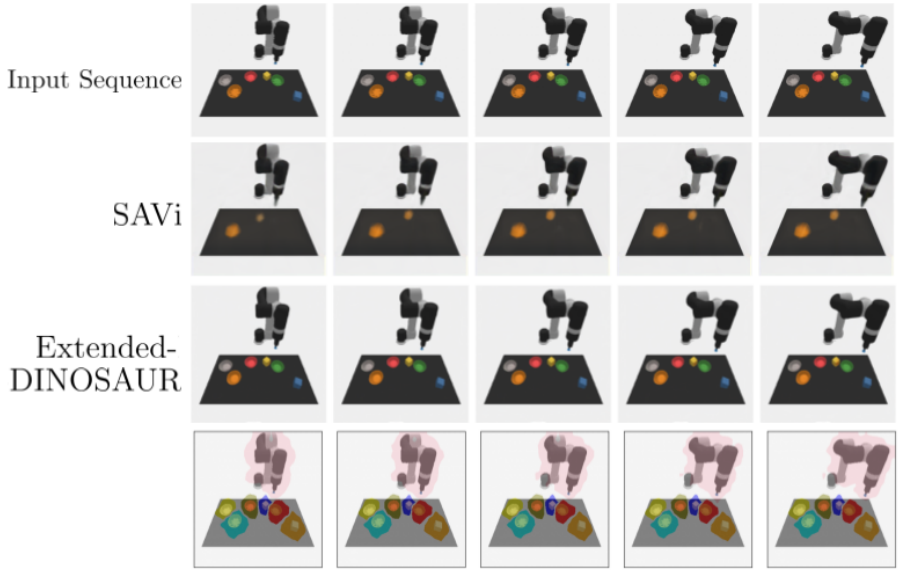


Figure 5.7: Reconstructed images using SAVi and Extended-DINOSAUR on CLIPort.

contributors to this limitation is the object-centric module that they use to learn object representations, i.e. SAVi. This inspired us to explore a new object-centric model and we propose Extended-DINOSAUR, which allowed our model to scale to more complex robotics datasets, such as CLIPort.

On Section 4.3.1, we reasoned why DINOSAUR presents a more suitable environment to learn object representations than SAVi. We also visualized the reconstructed images of pre-trained SAVi and Extended-DINOSAUR on CLIPort data, given a input video sequence of 5 frames. As observed in Figure 5.7, SAVi is not able to correctly model the objects on the 2D plane, while our proposed Extended-DINOSAUR, not only reconstructs the scene similar to the input, but its learned slots are able to model each individual object in the scene, as observed from the segmentation masks on the last row.

Furthermore, we note that we decided to use DINOv2 self-supervised method for pre-training the ViT encoder instead of DINO, since it resulted in slot representations that better described individual objects in the scene.

Number of Slots. The number of slots is an important hyper-parameter that is crucial in the model performance. We explore its impact on image-to-video generation task on CLIPort. Each scene on CLIPort can be described by 8 object representations (6 blocks/bowls + robot arm + background). We claim that having a few extra slots as buffers can improve the model performance.

To support this claim, we first pre-trained two identical Extended-DINOSAUR object-centric models, where one learns 8 slots per frame, while the other 10 slots per frame. Then, we trained the same predictor leveraging each one of the pre-trained object-centric models. As observed in Table 5.10, the predictor trained on the 10 slots per frame setting provides a better performance.

Table 5.10: Evaluation of the number of slots impact on model performance.

Number of slots	**CLIPort$_{1\rightarrow9}$**		
	PSNR	SSIM	LPIPS
8	25.951	0.890	0.079
10	**26.309**	**0.945**	**0.078**

5.3.2 Controllability

One of the main goals of our work on text-driven image-to-video generation, is to enable control over model predictions. This is achieved through the language instruction, which provides a description of objects and their anticipated motion. We analyze our model controllability abilities on both CATER-hard and CLIPort datasets.

CATER-hard In Figure 5.8, we provide an example of how our TextOCVP$_{\text{SAVi}}$ model adapts its prediction of future video frames based on the language instruction it receives as input. In the first two rows, we visualize the original ground truth video sequence from the dataset, its caption, and the prediction of our model, which receives as input only the first reference frame and the text caption. As can be seen, the generated frames closely follow the motion described on the text, and consequently, are very similar to the ground-truth frames.

We then change the input language instruction, by changing the actions being conducted on the objects, and pass the initial reference frame and this new sentence to our model. As can be observed on the third row of Figure 5.8, our model correctly adapts its prediction to express the new motion. This is not restricted only to changes on object actions, but we can input an entirely new prompt to the model. On the fourth row are visualized the predicted frames of TextOCVP$_{\text{SAVi}}$ when receiving as input a language instruction with changed objects and actions from the original prompt. Again, our model is able to generate frames that represent the new described motion.

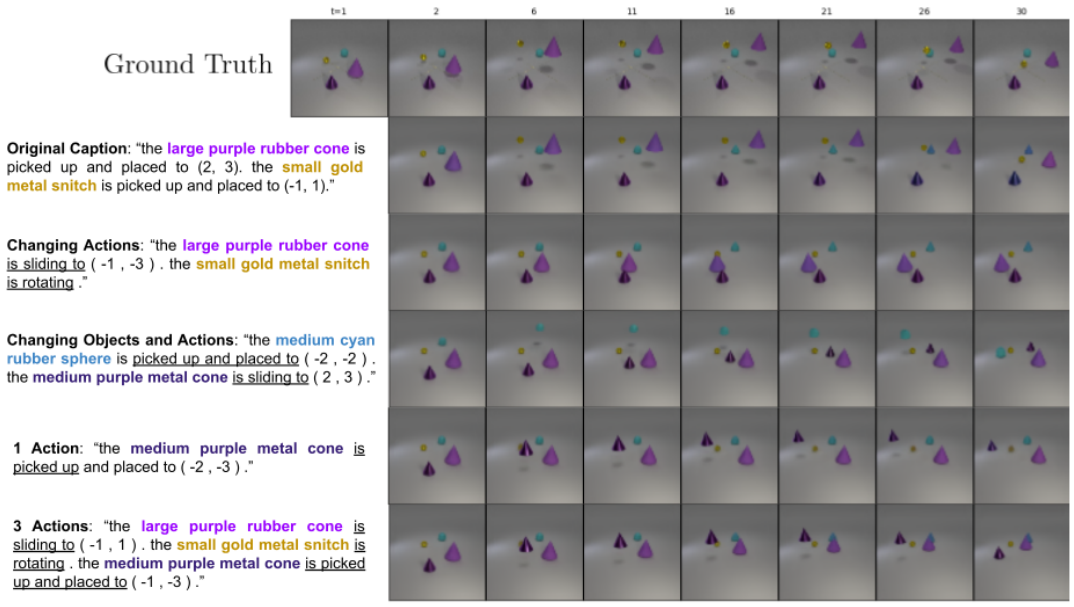


Figure 5.8: Examples of control over model's predictions by changing the language instruction.

On CATER-hard dataset, every video caption describes the motion of two objects in the scene. We experiment the model robustness to this bias, by proving the model input textual prompts that describe the motion of one or three objects. As visualized on the last two rows of Figure 5.8, our model is able to cope with these changes on the prompts' structure.

CLIPort We conduct a similar experiment on the CLIPort dataset, where we also compare the controllability ability of our TextOCVP model with $\text{MAGE}_{\text{DINO}}$'s. On Figure 5.9, we visualize the predicted sequence from both models, which receive as input the first seed frame and the video caption of the ground-truth video. Both models correctly express the motion described in the caption. However, in $\text{MAGE}_{\text{DINO}}$'s predictions, the cyan block changes colors and disappears after being put to the brown bowl, whereas the prediction of our model are more concise and the block is still visible inside the bowl. There is still some noise in the frames generated by our model, such as the color's change of the red bowl on top left of the table plane.

We then change the textual prompt, by instructing the models to put blue block in the brown bowl, instead of the cyan block. The predicted frames of both models are visualized on the last two rows of Figure 5.9. We can observe that $\text{MAGE}_{\text{DINO}}$ fails to adapt to this change. The robot arm still picks the cyan block, changes its

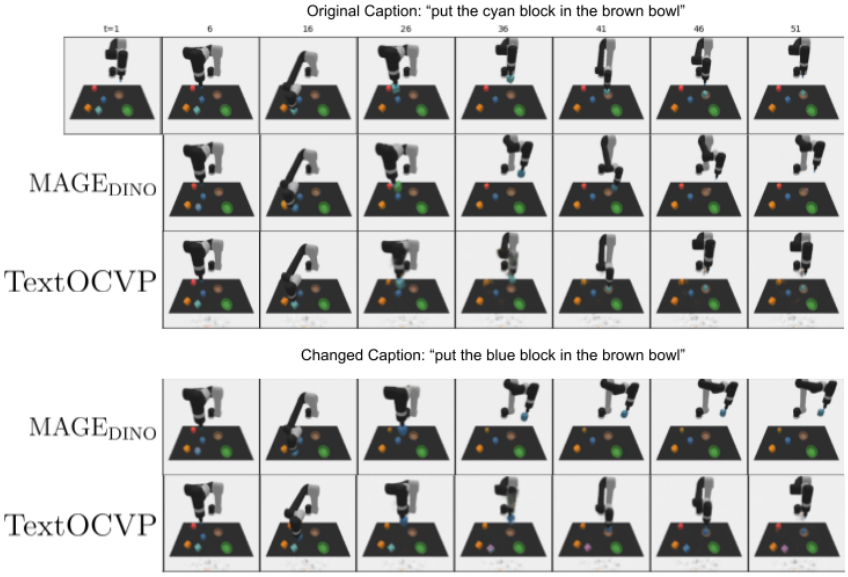


Figure 5.9: Comparing controllability abilities of TextOCVP and MAGE$_{DINO}$ on CLIPort.

color to blue, and is not able to put it on the brown bowl as it gets stuck. On the other hand, TextOCVP is able to correctly adapt to the new language instruction. The robot arm picks the blue block, and as instructed, puts it precisely on the brown bowl.

In general, the changing of the block in language instructions has proven difficult for both models. On the other hand, they were able to adapt easier to changes of the receiving bowl described in the prompt. We address this behavior to possible biases on the training dataset, where we have observed a lot of scenes with only a single block on the table.

5.3.3 Text-to-Slot Attention Visualizations

During our evaluation results, described in Section 5.2, we observed the importance of textual information on enhancing the ability of the model to correctly predict future frames. Particularly, on text-driven image-to-video generation task, the language instruction is crucial in guiding the predicted motion of objects in the scene, since we do not have any prior spatio-temporal information from the slots. Through visualizing the text-to-slot attention scores computed on the first predictor layer of our model, we attempt to interpret why the textual information is helpful.

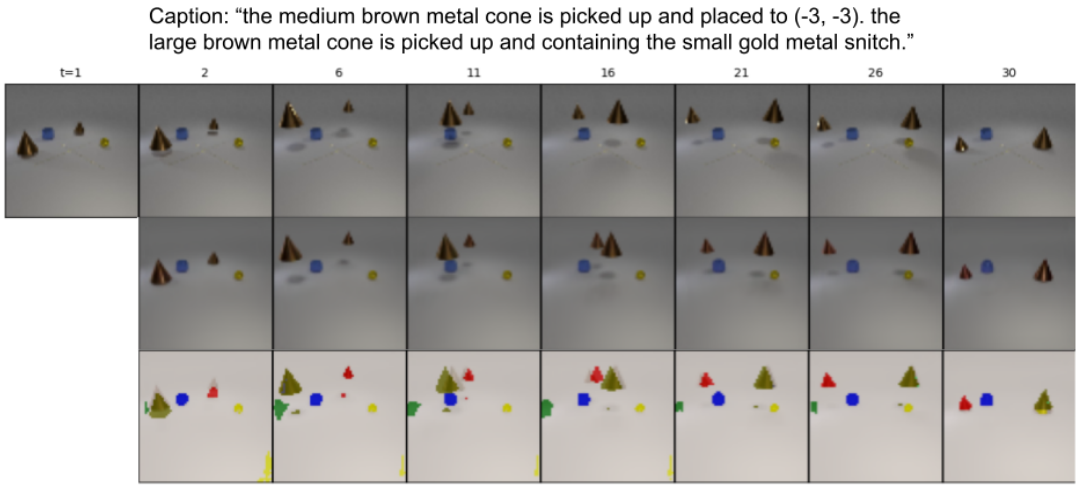


Figure 5.10: Example of predicted sequence and segmentation masks on CATER-hard, given a single seed frame and the language instruction.

In Figure 5.10, we illustrate the predicted sequence of TextOCVP$_{\text{SAVi}}$ on CATER-hard, given as input a seed frame and the language instruction. Our model correctly predicts the motion of both cones described in the instruction, despite them being the same color and both initially being picked up. We attribute this behavior to our object-centric approach and the language guidance.

As observed in the last row of Figure 5.10, where we visualize the segmentation received by the rendered and combined alpha masks from each slot, the object-centric model correctly learns representations for each individual object in the scene. Then, in the text-to-slot attention module, each slot representing an object included in the video caption, is enhanced by attending to important text tokens for that object. On Figure 5.11, we visualize the text-to-slot attention scores on the last head, between the two slots representing the cones and every text token. As can be observed, each slot attends more to the important information for the object that it represents, which initially is the ”picked up” action. Furthermore, despite being two such actions on the text prompt, the slot focuses only on the corresponding action of its object.

We also compute the text-to-slot attention scores on another sequence of CATER-hard, illustrated on Figure 5.12. The scene has two purple cones: one medium metal cone that is static, and a large rubber cone which should be picked up and placed in a specific coordinate. We visualize the text-to-slot attention score in the

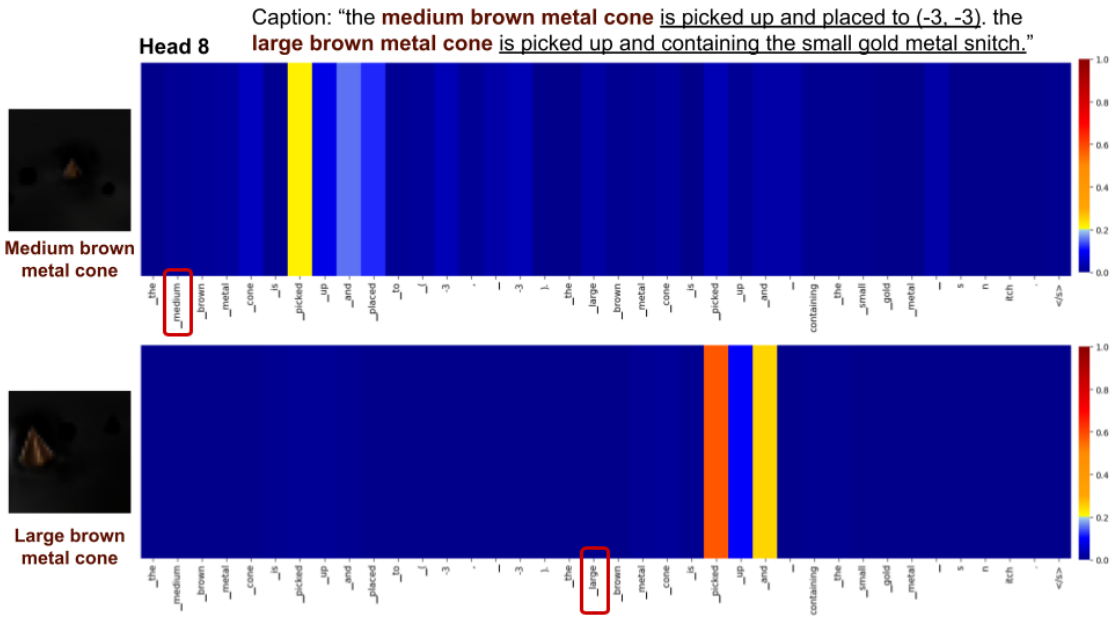


Figure 5.11: Text-to-slot attention scores on head 8, between the slots representing the two brown cones and every text token.

first time step, computed on the slots that represent each cone. For simplicity, we visualize the scores computed on head 3 and 8, which proved to be more informative in this scenario. As can be observed in Figure 5.13, the attention scores computed on the slot that represents the large purple rubber cone are high on tokens that represent its attributes, such as 'purple' and 'rubber', and anticipated action, i.e. "picked". On the other hand, the slot that represents the static cone does not attend more to a particular important text token, but rather to tokens like "the".

5.3.4 Model Robustness

Unseen Colors. As described in Section 5.1.1, CLIPort's vocabulary is limited, and the colors of the blocks and bowls mentioned in the language instructions within the training and validation sets are constrained to a small set.

We evaluate the performance of our TextOCVP model and $\text{MAGE}_{\text{DINO}}$ on the CLIPort test set, which includes color variations in the text instructions that were not encountered during training. As shown in Table 5.11, our model outperforms $\text{MAGE}_{\text{DINO}}$ when predicting 9 and 19 future frames. Furthermore, through a comparison with the evaluation results on the training set, presented in Table 5.4, we can observe that $\text{MAGE}_{\text{DINO}}$ is more significantly impacted by changes in color.

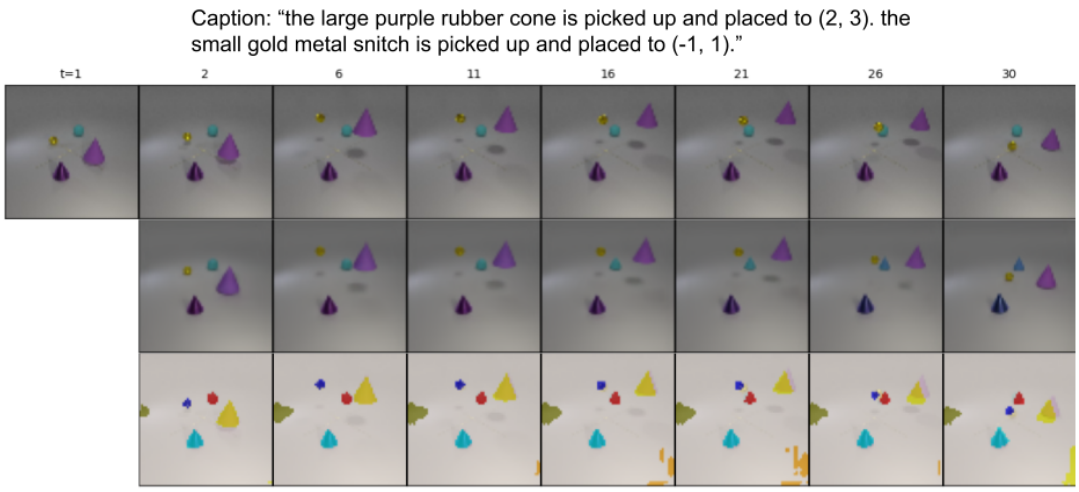


Figure 5.12: Example of another predicted sequence and segmentation masks on CATER-hard, given a single seed frame and the language instruction.

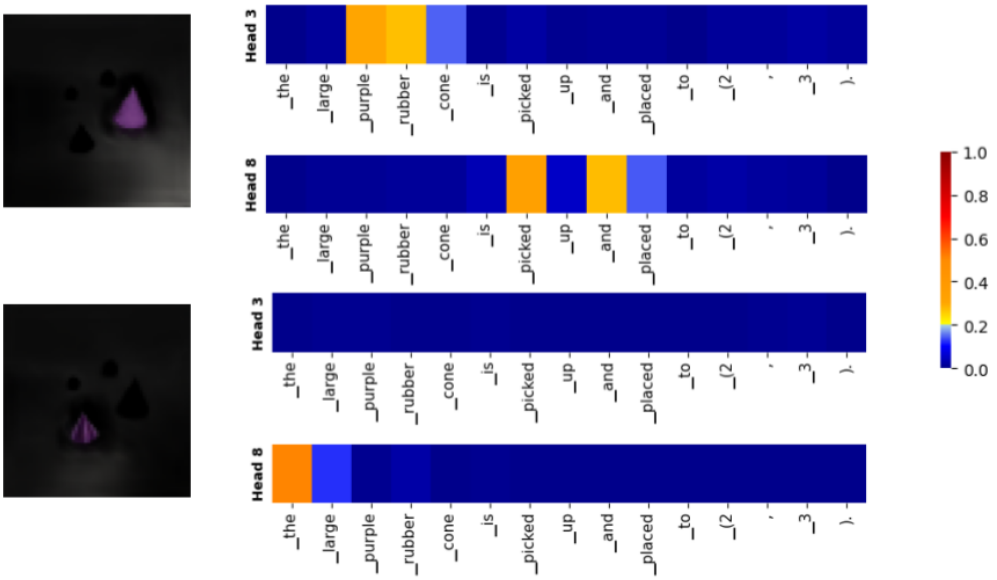


Figure 5.13: Text-to-slot attention scores on heads 3 and 8, between the slots representing the two purple cones and every text token.

Both models maintain a high SSIM value, as the structure of the objects in the scene remains unchanged. However, $\text{MAGE}_{\text{DINO}}$ shows a notable performance drop of over 16% on the LPIPS metric, compared to only 6.4% for TextOCVP.

Table 5.11: Quantitative evaluation on CLIPort test set.

Method	$\textbf{CLIPort}_{1\to9}$			$\textbf{CLIPort}_{1\to19}$		
	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
$\textbf{MAGE}_{\textbf{DINO}}$	22.907 (-3.4%)	0.935 (-0.5%)	0.076 (-18.7%)	21.361 (-4.1%)	0.924 (-0.7%)	0.087 (-16.0%)
TextOCVP	**26.158** (-3.1%)	**0.946** (-0.4%)	**0.066** (-6.4%)	**23.168** (-3.0%)	**0.927** (-0.4%)	**0.083** (-6.4%)

This lower difference in the visual quality of the generated frames, demonstrates that TextOCVP is more robust and generalizes better to unseen colors.

6 Conclusion and Future Work

6.1 Conclusion

In this thesis, we presented TextOCVP, a novel object-centric model for text-driven image-to-video generation. To address the limitations of previous object-centric approaches, we introduced a new text-conditioned predictor, which integrates textual information to guide predictions. Furthermore, we adapted a new object-centric module capable of learning object representations in datasets with complex object dynamics.

The proposed model can take a single input image and a textual instruction as input. Using the object-centric module, it first extracts object representations from the image. Then, the text-conditioned predictor models the spatio-temporal relationships between these objects while incorporating the guidance provided by the textual instruction. Finally, the object-centric module's decoder generates video frames that reflect the described motion, enabling the creation of high-quality video sequences aligned with the language input.

Through extensive evaluations on various datasets, we demonstrated that TextOCVP outperforms existing text-conditioned models for the image-to-video generation task. Quantitative and qualitative results highlighted the model's ability to generate visually consistent long sequences. We conduct various ablation studies to support and validate the architectural choices for our model components. Additionally, we demonstrated the model's ability to adapt its predictions based on the objects and motions described in the language input, highlighting the importance of combining textual and object-centric information.

This work demonstrates the benefits of object-centricity in video generation and the advantages of integrating language guidance to create flexible and interpretable models for future scene generation. By bridging the fields of object-centric representation learning and language-guided video generation, this research provides a solid foundation for future work. It has potential applications in robotics, autonomous systems, and other domains that require controllable video generation based on textual instructions.

6.2 Future Work

While our work has made significant contributions to text-driven image-to-video generation following an object-centric approach, there remain several open challenges and promising directions for future exploration.

One important area for improvement is the application to real-world datasets. Although our proposed object-centric module has shown potential for scaling to more complex datasets, further research is needed to fully bridge the gap between synthetic and real-world video data. Advancements in object-centric learning, such as improved visual feature extraction, more effective decoders, and novel training objectives, could significantly enhance performance. Combining these approaches with methods, such as vector-quantized representations or diffusion models, may also lead to more realistic video generation.

Another critical direction is the evaluation of model robustness in text-driven image-to-video generation. Future work should explore how well models handle ambiguity in language descriptions and assess their ability to generalize to unseen scenes or settings that differ from the training environment. Such evaluations would provide deeper insights into the reliability and adaptability of these models, particularly in practical applications.

Finally, we believe that integrating object-centric approaches with text-driven video generation presents a promising research direction for the future. This combination has the potential to not only improve model performance but also enable scaling to real-world scenarios. By addressing the outlined challenges, this research area can contribute to advancements in various domains, including robotics, autonomous systems, and interactive AI applications.

List of Figures

List of Tables

Bibliography

Ba, Lei Jimmy, Jamie Ryan Kiros, and Geoffrey E. Hinton (2016). "Layer normalization." In: *Corr*.

Bahdanau, Dzmitry, Kyunghyun Cho, and Yoshua Bengio (2015). "Neural machine translation by jointly learning to align and translate." In: *3rd international conference on learning representations, ICLR 2015, san diego, ca, usa, may 7-9, 2015, conference track proceedings*.

Bengio, Yoshua, Aaron Courville, and Pascal Vincent (2013). "Representation learning: a review and new perspectives." In: *Ieee transactions on pattern analysis and machine intelligence*.

Burgess, Christopher P., Loïc Matthey, Nicholas Watters, Rishabh Kabra, Irina Higgins, Matthew M. Botvinick, and Alexander Lerchner (2019). "Monet: unsupervised scene decomposition and representation." In: *Corr*.

Caron, Mathilde, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin (2020). "Unsupervised learning of visual features by contrasting cluster assignments." In: *Advances in neural information processing systems*.

Caron, Mathilde, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin (2021). "Emerging properties in self-supervised vision transformers." In: *Proceedings of the ieee/cvf international conference on computer vision*.

Cheng, Jianpeng, Li Dong, and Mirella Lapata (2016). "Long short-term memory-networks for machine reading." In: *Proceedings of the 2016 conference on empirical methods in natural language processing*.

Chiu, Hsu-Kuang, Ehsan Adeli, and Juan Carlos Niebles (2020). "Segmenting the future." In: *Ieee robotics and automation letters*.

Cho, Kyunghyun, Bart van Merrienboer, Çaglar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio (2014). "Learning phrase representations using RNN encoder-decoder for statistical machine translation." In: *Proceedings of the 2014 conference on empirical methods in natural language processing, EMNLP 2014, october 25-29, 2014, doha, qatar, A meeting of sigdat, a special interest group of the ACL*.

Creswell, Antonia, Rishabh Kabra, Christopher P. Burgess, and Murray Shanahan (2021). "Unsupervised object-based transition models for 3d partially observable environments." In: *Advances in neural information processing systems 34: annual conference on neural information processing systems 2021, neurips 2021, december 6-14, 2021, virtual*.

Cybenko, George (1989). "Approximation by superpositions of a sigmoidal function." In: *Mathematics of control, signals and systems*.

Daniel, Tal and Aviv Tamar (2024). "DDLP: Unsupervised Object-centric Video Prediction with Deep Dynamic Latent Particles." In: *Transactions on machine learning research (tmlr)*.

Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova (2019). "BERT: pre-training of deep bidirectional transformers for language understanding." In: *Proceedings of the 2019 conference of the north american chapter of the association for computational linguistics: human language technologies, NAACL-HLT 2019, minneapolis, mn, usa, june 2-7, 2019, volume 1 (long and short papers)*.

Elsayed, Gamaleldin, Aravindh Mahendran, Sjoerd Van Steenkiste, Klaus Greff, Michael C Mozer, and Thomas Kipf (2022). "Savi++: towards end-to-end object-centric learning from real-world videos." In: *Advances in neural information processing systems*.

Farazi, Hafez, Jan Nogga, and Sven Behnke (2021). "Local frequency domain transformer networks for video prediction." In: *International joint conference on neural networks, IJCNN 2021, shenzhen, china, july 18-22, 2021*.

Gao, Zhangyang, Cheng Tan, Lirong Wu, and Stan Z. Li (2022). "Simvp: simpler yet better video prediction." In: *IEEE/CVF conference on computer vision and pattern recognition, CVPR 2022, new orleans, la, usa, june 18-24, 2022*. IEEE.

Girdhar, Rohit and Deva Ramanan (2020). "CATER: A diagnostic dataset for compositional actions & temporal reasoning." In: *8th international conference on learning representations, ICLR 2020, addis ababa, ethiopia, april 26-30, 2020*.

Goodfellow, Ian (2016). *Deep learning*.

Goyal, Raghav, Samira Ebrahimi Kahou, Vincent Michalski, Joanna Materzynska, Susanne Westphal, Heuna Kim, Valentin Haenel, Ingo Fründ, Peter Yianilos, Moritz Mueller-Freitag, Florian Hoppe, Christian Thurau, Ingo Bax, and Roland Memisevic (2017). "The "something something" video database for learning and evaluating visual common sense." In: *IEEE international conference on computer vision, ICCV 2017, venice, italy, october 22-29, 2017*.

Greff, Klaus, Raphaël Lopez Kaufman, Rishabh Kabra, Nick Watters, Chris Burgess, Daniel Zoran, Loic Matthey, Matthew M. Botvinick, and Alexander Lerchner (2019). "Multi-object representation learning with iterative variational inference." In: *Proceedings of the 36th international conference on machine learning, ICML 2019, 9-15 june 2019, long beach, california, USA*.

Gu, Xianfan, Chuan Wen, Weirui Ye, Jiaming Song, and Yang Gao (2024). "Seer: language instructed video prediction with latent diffusion models." In: *The twelfth international conference on learning representations, ICLR 2024, vienna, austria, may 7-11, 2024*.

Guen, Vincent Le and Nicolas Thome (2020). "Disentangling physical dynamics from unknown factors for unsupervised video prediction." In: *2020 IEEE/CVF*

conference on computer vision and pattern recognition, CVPR 2020, seattle, wa, usa, june 13-19, 2020.

Haykin, Simon (1998). *Neural networks: a comprehensive foundation*. Prentice Hall PTR.

He, Kaiming, Xiangyu Zhang, Shaoqing Ren, and Jian Sun (2016). "Deep residual learning for image recognition." In: *2016 IEEE conference on computer vision and pattern recognition, CVPR 2016, las vegas, nv, usa, june 27-30, 2016*.

Hendrycks, Dan and Kevin Gimpel (2016). "Gaussian error linear units (gelus)." In: *Arxiv preprint arxiv:1606.08415*.

Hornik, Kurt, Maxwell Stinchcombe, and Halbert White (1989). "Multilayer feedforward networks are universal approximators." In: *Neural networks*.

Hu, Yaosi, Chong Luo, and Zhenzhong Chen (2022). "Make it move: controllable image-to-video generation with text descriptions." In: *IEEE/CVF conference on computer vision and pattern recognition, CVPR 2022, new orleans, la, usa, june 18-24, 2022*.

Kingma, Diederik P. and Jimmy Ba (2015). "Adam: A method for stochastic optimization." In: *3rd international conference on learning representations, ICLR 2015, san diego, ca, usa, may 7-9, 2015, conference track proceedings*. Ed. by Yoshua Bengio and Yann LeCun.

Kipf, Thomas, Gamaleldin Fathy Elsayed, Aravindh Mahendran, Austin Stone, Sara Sabour, Georg Heigold, Rico Jonschkowski, Alexey Dosovitskiy, and Klaus Greff (2022). "Conditional object-centric learning from video." In: *The tenth international conference on learning representations, ICLR 2022, virtual event, april 25-29, 2022*.

Kolesnikov, Alexander, Alexey Dosovitskiy, Dirk Weissenborn, Georg Heigold, Jakob Uszkoreit, Lucas Beyer, Matthias Minderer, Mostafa Dehghani, Neil Houlsby, Sylvain Gelly, Thomas Unterthiner, and Xiaohua Zhai (2021). "An image is worth 16x16 words: transformers for image recognition at scale." In: *9th international conference on learning representations, ICLR 2021, virtual event, austria, may 3-7, 2021*.

Krizhevsky, Alex (2014). "One weird trick for parallelizing convolutional neural networks." In: *Corr*.

Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E Hinton (2012). "Imagenet classification with deep convolutional neural networks." In: *Advances in neural information processing systems*.

LeCun, Yann, Yoshua Bengio, et al. (1995). "Convolutional networks for images, speech, and time series." In: *The handbook of brain theory and neural networks*.

LeCun, Yann, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel (1989). "Backpropagation applied to handwritten zip code recognition." In: *Neural computation*.

LeCun, Yann, Léon Bottou, Yoshua Bengio, and Patrick Haffner (1998). "Gradient-based learning applied to document recognition." In: *Proceedings of the ieee*.

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: *8th international conference on learning representations, ICLR 2020, addis ababa, ethiopia, april 26-30, 2020*.

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 33: annual conference on neural information processing systems 2020, neurips 2020, december 6-12, 2020, virtual*.

Luong, Thang, Hieu Pham, and Christopher D. Manning (2015). "Effective approaches to attention-based neural machine translation." In: *Proceedings of the 2015 conference on empirical methods in natural language processing*.

Oquab, Maxime, Timothée Darcet, Théo Moutakanni, Huy V. Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, Mahmoud Assran, Nicolas Ballas, Wojciech Galuba, Russell Howes, Po-Yao Huang, Shang-Wen Li, Ishan Misra, Michael G. Rabbat, Vasu Sharma, Gabriel Synnaeve, Hu Xu, Hervé Jégou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski (2023). "Dinov2: learning robust visual features without supervision." In: *Corr*.

Radford, Alec, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever (2021). "Learning transferable visual models from natural language supervision." In: *Proceedings of the 38th international conference on machine learning, ICML 2021, 18-24 july 2021, virtual event*.

Raffel, Colin, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu (2020). "Exploring the limits of transfer learning with a unified text-to-text transformer." In: *Journal of machine learning research*.

Rakhimov, Ruslan, Denis Volkhonskiy, Alexey Artemov, Denis Zorin, and Evgeny Burnaev (2021). "Latent video transformer." In: *Proceedings of the 16th international joint conference on computer vision, imaging and computer graphics theory and applications, VISIGRAPP 2021, volume 5: visapp, online streaming, february 8-10, 2021*.

Rombach, Robin, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer (2022). "High-resolution image synthesis with latent diffusion models." In: *IEEE/CVF conference on computer vision and pattern recognition, CVPR 2022, new orleans, la, usa, june 18-24, 2022*.

Rumelhart, David E, Geoffrey E Hinton, and Ronald J Williams (1986). "Learning representations by back-propagating errors." In: *Nature*.

Russakovsky, Olga, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael S. Bernstein,

Alexander C. Berg, and Li Fei-Fei (2015). "Imagenet large scale visual recognition challenge." In: *International journal of computer vision.*

Sampat, Mehul P., Zhou Wang, Shalini Gupta, Alan Conrad Bovik, and Mia K. Markey (2009). "Complex wavelet structural similarity: A new image similarity index." In: *IEEE transactions image processing.*

Sara, Umme, Morium Akter, and Mohammad Shorif Uddin (2019). "Image quality assessment through fsim, ssim, mse and psnr—a comparative study." In: *Journal of computer and communications.*

Seitzer, Maximilian, Max Horn, Andrii Zadaianchuk, Dominik Zietlow, Tianjun Xiao, Carl-Johann Simon-Gabriel, Tong He, Zheng Zhang, Bernhard Schölkopf, Thomas Brox, and Francesco Locatello (2023). "Bridging the gap to real-world object-centric learning." In: *The eleventh international conference on learning representations, ICLR 2023, kigali, rwanda, may 1-5, 2023.*

Shridhar, Mohit, Lucas Manuelli, and Dieter Fox (2021). "Cliport: what and where pathways for robotic manipulation." In: *Conference on robot learning, 8-11 november 2021, london, UK.*

Simonyan, Karen and Andrew Zisserman (2015). "Very deep convolutional networks for large-scale image recognition." In: *3rd international conference on learning representations, ICLR 2015, san diego, ca, usa, may 7-9, 2015, conference track proceedings.*

Singh, Gautam, Yi-Fu Wu, and Sungjin Ahn (2022). "Simple unsupervised object-centric learning for complex and naturalistic videos." In: *Advances in neural information processing systems (neurips).*

Song, Xue, Jingjing Chen, Bin Zhu, and Yu-Gang Jiang (2024). "Text-driven video prediction." In: *Acm transactions on multimedia computing, communications and applications.*

Touvron, Hugo, Andrea Vedaldi, Matthijs Douze, and Hervé Jégou (2019). "Fixing the train-test resolution discrepancy." In: *Advances in neural information processing systems.*

Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin (2017). "Attention is all you need." In: *Advances in neural information processing systems 30: annual conference on neural information processing systems 2017, december 4-9, 2017, long beach, ca, USA*. Ed. by Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett.

Villar-Corrales, Angel and Sven Behnke (2022). "Unsupervised image decomposition with phase-correlation networks." In: *Proceedings of the 17th international joint conference on computer vision, imaging and computer graphics theory and applications, VISIGRAPP 2022, volume 4: visapp, online streaming, february 6-8, 2022.*

Villar-Corrales, Angel, Ani Karapetyan, Andreas Boltres, and Sven Behnke (2022). "Mspred: video prediction at multiple spatio-temporal scales with hierarchical

recurrent networks." In: *33rd british machine vision conference 2022, BMVC 2022, london, uk, november 21-24, 2022*.

Villar-Corrales, Angel, Ismail Wahdan, and Sven Behnke (2023). "Object-centric video prediction via decoupling of object dynamics and interactions." In: *IEEE international conference on image processing, ICIP 2023, kuala lumpur, malaysia, october 8-11, 2023*.

Wang, Yunbo, Haixu Wu, Jianjin Zhang, Zhifeng Gao, Jianmin Wang, Philip S. Yu, and Mingsheng Long (2023). "Predrnn: A recurrent neural network for spatiotemporal predictive learning." In: *IEEE trans. pattern anal. mach. intell.*

Wang, Zhou, Alan C. Bovik, Hamid R. Sheikh, and Eero P. Simoncelli (2004). "Image quality assessment: from error visibility to structural similarity." In: *IEEE transactions image processing*.

Watters, Nicholas, Loïc Matthey, Christopher P. Burgess, and Alexander Lerchner (2019). "Spatial broadcast decoder: A simple architecture for learning disentangled representations in vaes." In: *Corr*.

Wu, Yi-Fu, Jaesik Yoon, and Sungjin Ahn (2021). "Generative video transformer: can objects be the words?" In: *Proceedings of the 38th international conference on machine learning, ICML 2021, 18-24 july 2021, virtual event*.

Wu, Ziyi, Nikita Dvornik, Klaus Greff, Thomas Kipf, and Animesh Garg (2023). "Slotformer: unsupervised visual dynamics simulation with object-centric models." In: *The eleventh international conference on learning representations, ICLR 2023, kigali, rwanda, may 1-5, 2023*.

Ye, Yufei, Maneesh Singh, Abhinav Gupta, and Shubham Tulsiani (2019). "Compositional video prediction." In: *2019 IEEE/CVF international conference on computer vision, ICCV 2019, seoul, korea (south), october 27 - november 2, 2019*.

Zadaianchuk, Andrii, Maximilian Seitzer, and Georg Martius (2024). "Object-centric learning for real-world videos by predicting temporal feature similarities." In: *Advances in neural information processing systems (neurips)*.

Zeiler, MD (2014). "Visualizing and understanding convolutional networks." In: *European conference on computer vision/arxiv*.

Zhang, Richard, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang (2018). "The unreasonable effectiveness of deep features as a perceptual metric." In: *2018 IEEE conference on computer vision and pattern recognition, CVPR 2018, salt lake city, ut, usa, june 18-22, 2018*.