STRUCTURED OBJECT-AWARE PHYSICS PREDICTION FOR VIDEO MODELING AND PLANNING

Jannik Kossen[*1,2], **Karl Stelzner**[*2], **Marcel Hussing**[2], **Claas Voelcker**[2], **Kristian Kersting**[2,3]
[1]Department of Physics and Astronomy, Heidelberg University, Germany
[2]Department of Computer Science, [3] Centre for Cognitive Science, TU Darmstadt, Germany
kossen@stud.uni-heidelberg.de, {stelzner,kersting}@cs.tu-darmstadt.de
{marcel.hussing,c.voelcker}@stud.tu-darmstadt.de

ABSTRACT

When humans observe a physical system, they can easily locate objects, understand their interactions, and anticipate future behavior, even in settings with complicated and previously unseen interactions. For computers, however, learning such models from videos in an unsupervised fashion is an unsolved research problem. In this paper, we present STOVE, a novel state-space model for videos, which explicitly reasons about objects and their positions, velocities, and interactions. It is constructed by combining an image model and a dynamics model in compositional manner and improves on previous work by reusing the dynamics model for inference, accelerating and regularizing training. STOVE predicts videos with convincing physical behavior over hundreds of timesteps, outperforms previous unsupervised models, and even approaches the performance of supervised baselines. We further demonstrate the strength of our model as a simulator for sample efficient model-based control in a task with heavily interacting objects.

1 Introduction

Obtaining structured knowledge about the world from unstructured, noisy sensory input is a key challenge in artificial intelligence. Of particular interest is the problem of identifying objects from visual input and understanding their interactions. One longstanding approach to this is the idea of *vision as inverse graphics* (Grenander, 1976), which postulates a data generating graphics process and phrases vision as posterior inference in the induced distribution. Despite its intuitive appeal, it has remained largely intractable in practice due to the high-dimensional and multimodal nature of the inference problem. Recently, however, probabilistic models based on deep neural networks have made promising advances in this area. By composing conditional distributions parameterized by neural networks, highly expressive yet structured models have been built. At the same time, advances in general approximate inference, particularly variational techniques, have put the inference problem for these models within reach (Zhang et al., 2018).

Based on these advances, a number of probabilistic models for unsupervised scene understanding in single images have recently been proposed. The structured nature of approaches such as AIR (Eslami et al., 2016) and MONet (Burgess et al., 2019) provides two key advantages over unstructured image models such as variational autoencoders (Kingma and Welling, 2014) or generative adversarial networks (Goodfellow et al., 2014). First, it allows for the specification of inductive biases, such as spatial consistency of objects, which constrain the model and act as regularization. Second, it enables the use of semantically meaningful latent variables, such as object positions, which may be used for downstream reasoning tasks.

Building such a structured model for videos instead of individual images is the natural next challenge. Not only could such a model be used in more complex domains, such as reinforcement learning, but the additional redundancy in the data can even simplify and regularize the object detection problem (Kosiorek et al., 2018). To this end, the notion of temporal consistency may be leveraged as an additional inductive bias, guiding the model to desirable behavior. In situations where interactions between objects are prevalent, understanding and explicitly modeling these interactions in an object-centric state-space is valuable for obtaining good predictive models (Watters et al., 2017). Existing works in this area, such as SQAIR (Kosiorek et al., 2018), DDPAE (Hsieh et al., 2018), R-NEM (Van Steenkiste et al., 2018),

[*]Equal contribution

and COBRA (Watters et al., 2019) have explored these concepts, but have not demonstrated realistic long term video predictions on par with supervised approaches to modeling physics.

To push the limits of unsupervised learning of physical interactions, we propose *STOVE*, a structured, object-aware video model. With STOVE, we combine image and physics modeling into a single state-space model, which explicitly reasons about object positions and velocities. It is trained end-to-end on pure video data in a self-supervised fashion and learns to detect objects, to model their interactions, and to predict future states and observations. To facilitate learning via variational inference in this model, we provide a novel inference architecture, which reuses the learned generative physics model in the variational distribution. As we will demonstrate, our model generates convincing rollouts over hundreds of time steps, outperforms other video modeling approaches, and in fact approaches the performance of the supervised baseline which has access to the ground truth object states.

Moving beyond unsupervised learning, we also demonstrate how STOVE can be employed for model-based reinforcement learning (RL). Model-based approaches to RL have long been viewed as a potential remedy to the often prohibitive sample complexity of model-free RL, but obtaining learned models of sufficient quality has proven difficult in practice (Sutton and Barto, 2011). By conditioning state predictions on actions and adding reward predictions to our dynamics predictor, we extend our model to the RL setting, allowing it to be used for search or planning. Our empirical evidence shows that an actor based on Monte-Carlo tree search (MCTS) (Coulom, 2007) on top of our model is competitive to model-free approaches such as Proximal Policy Optimization (PPO) (Schulman et al., 2017), while only requiring a fraction of the samples.

We proceed by introducing the two main components of STOVE: a structured image model and a dynamics model. We show how to perform joint inference and training, as well as how to extend the model to the RL setting. We then present our experimental evaluation, before touching on further related work and concluding.

2 Structured Object-Aware Video Modeling

We approach the task of modeling a video with frames $x_1, \dots, x_T$ from a probabilistic perspective, assuming a sequence of Markovian latent states $z_1, \dots, z_T$, which decompose into the properties of a fixed number O of objects, i.e. $z_t = (z_t^1, \dots, z_t^O)$. In the spirit of compositionality, we propose to specify and train such a model by explicitly combining a dynamics prediction model $p(z_{t+1} \mid z_t)$ and a scene model $p(x_t \mid z_t)$. This yields a state-space model, which can be trained on pure video data, using variational inference and an approximate posterior distribution $q(z \mid x)$. Our model differs from previous work that also follows this methodology, most notably SQAIR and DDPAE, in three major ways:

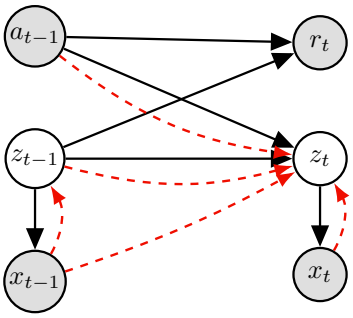
- We propose a more compact architecture for the variational distribution $q(z \mid x)$, which reuses the dynamics model $p(z_{t+1} \mid z_t)$, and avoids the costly double recurrence across time and objects which was present in previous work.
- We parameterize the dynamics model using a graph neural network, taking advantage of the decomposed nature of the latent state z.
- Instead of treating each z_t^o as an arbitrary latent code, we explicitly reserve the first six slots of this vector for the object's position, size, and velocity, each in x, y direction, and use this information for the dynamics prediction task. We write $z_t^o = (z_{t,\text{pos}}^o, z_{t,\text{size}}^o, z_{t,\text{velo}}^o, z_{t,\text{latent}}^o)$.

We begin by briefly introducing the individual components before discussing how they are combined to form our state-space model. An overview of the graphical model is given in Fig. 1 (left).

2.1 Object-based Modeling of Images using Sum-Product Attend-Infer-Repeat

A variety of object-centric image models have recently been proposed, many of which are derivatives of attend-infer-repeat (AIR) (Eslami et al., 2016). AIR postulates that each image consists of a set of objects, each of which occupies a rectangular region in the image, specified by positional parameters $z_{\text{where}}^o = (z_{\text{pos}}^o, z_{\text{size}}^o)$. The visual content of each object is described by a latent code z_{what}^o. By decoding z_{what}^o with a neural network and rendering the resulting image patches in the prescribed location, a generative model $p(x \mid z)$ is obtained. Inference is accomplished using a recurrent neural network, which outputs distributions over the latent objects $q(z^o \mid x)$, attending to one object at a time. AIR is also capable of handling varying numbers of objects, using an additional set of latent variables.

Sum-Product Attend-Infer-Repeat (SuPAIR) (Stelzner et al., 2019) utilizes sum-product networks (SPNs) instead of a decoder network to directly model the distribution over object appearances. The tractable inference capabilities of the SPNs used in SuPAIR allow for the exact and efficient computation of $p(x \mid z_{\text{where}})$, effectively integrating out the



$z^o_{t,\text{pos}}$	$q(z^o_{t,\text{pos}} \mid z_{t-1}) \cdot q(z^o_{t,\text{pos}} \mid x_t)$
$z^o_{t,\text{velo}}$	$q(z^o_{t,\text{velo}} \mid z_{t-1}) \cdot q(z^o_{t,\text{velo}} \mid x_t, x_{t-1})$
$z^o_{t,\text{size}}$	$q(z^o_{t,\text{size}} \mid x_t)$
$z^o_{t,\text{latent}}$	$q(z^o_{t,\text{latent}} \mid z_{t-1})$

Figure 1: (Left) Depiction of the graphical model underlying STOVE. Black arrows denote the generative mechanism and red arrows the inference procedure. The variational distribution $q(z_t \mid z_{t-1}, x_t, x_{t-1})$ is formed by combining predictions from the dynamics model $p(z_t \mid z_{t-1})$ and the object detection network $q(z_t \mid x_t)$. For the RL domain, our approach is extended by action conditioning and reward prediction. (Right) Components of z^o_t and corresponding variational distributions. Note that the velocities are estimated based on the change in positions between timesteps, inducing a dependency on x_{t-1}.

appearance parameters z_{what} analytically. This has been shown to drastically accelerate learning, as the reduced inference workload significantly lowers the variance of the variational objective. Since the focus of SuPAIR on interpretable object parameters fits our goal of building a structured video model, we apply it as our image model $p(x_t \mid z_t)$. Similarly, we use an inference network as in SuPAIR to model $q(z_{t,\text{where}} \mid x_t)$. For details on SuPAIR, we refer to Stelzner et al. (2019).

2.2 Modeling Physical Interactions using Graph Networks

In order to successfully capture complex dynamics, the state transition distribution $p(z_{t+1} \mid z_t) = p(z^1_{t+1}, \dots, z^O_{t+1} \mid z^1_t, \dots, z^O_t)$ needs to be parameterized using a flexible, non-linear estimator. A critical property that should be maintained in the process is *permutation invariance*, i.e., the output should not depend on the order in which objects appear in the vector z_t. This type of function is well captured by graph neural networks, cf. (Santoro et al., 2017), which posit that the output should depend on the sum of pairwise interactions between objects. Graph neural networks have been extensively used for modeling physical processes in supervised scenarios (Battaglia et al., 2016, 2018; Sanchez-Gonzalez et al., 2018; Zhou et al., 2018).

Following this line of work, we build a dynamics model of the basic form

$$\hat{z}^o_{t+1,\text{pos}}, \hat{z}^o_{t+1,\text{velo}}, \hat{z}^o_{t+1,\text{latent}} = f\left(g(z^o_t) + \sum_{o' \neq o} \alpha(z^o_t, z^{o'}_t) h(z^o_t, z^{o'}_t)\right) \tag{1}$$

where f, g, h, α represent functions parameterized by dense neural networks. α is an attention mechanism outputting a scalar which allows the network to focus on specific object pairs. We assume a constant prior over the object sizes, i. e., $\hat{z}^o_{t+1,\text{size}} = z^o_{t,\text{size}}$. The full state transition distribution is then given by the Gaussian $p(z^o_{t+1} \mid z^o_t) = \mathcal{N}(\hat{z}^o_{t+1}, \sigma)$, using a fixed σ.

2.3 Joint State-Space Model

Next, we assemble a state-space model from the two separate, compositional models for image modeling and physics prediction. The interface between the two component models are the latent positions and velocities. The scene model infers them from images and the physics model propagates them forward in time. Combining the two yields the state-space model $p(x, z) = p(z_0)p(x_0 \mid z_0) \prod_t p(z_t \mid z_{t-1})p(x_t \mid z_t)$. To initialize the state, we model $p(z_0, z_1)$ using simple uniform and Gaussian distributions. Details are given in Appendix B.3.

Our model is trained on given video sequences x by maximizing the evidence lower bound (ELBO) $\mathbb{E}_{q(z|x)}\left[\log p(x, z) - \log q(z \mid x)\right]$. This requires formulating a variational distribution $q(z \mid x)$ to approximate the true posterior $p(z \mid x)$. A natural approach is to factorize this distribution over time, i.e. $q(z \mid x) = q(z_0 \mid x_0) \prod_t q(z_t \mid z_{t-1}, x_t)$, resembling a Bayesian filter. The distribution $q(z_0 \mid x_0)$ is then readily available using the inference network provided by SuPAIR.

The formulation of $q(z_t \mid z_{t-1}, x_t)$, however, is an important design decision. Previous work, including SQAIR and DDPAE, have chosen to unroll this distribution over objects, introducing a costly double recurrence over time and

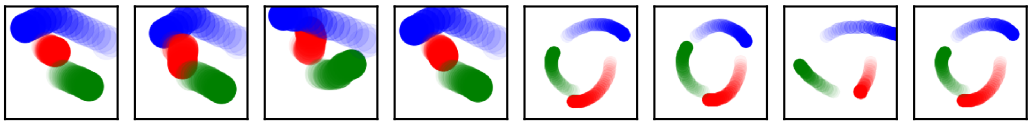


Figure 2: Visualisation of predicted object positions. Each illustrations shows object positions from the real environment, predictions made by our model, SQAIR, and the supervised baseline, after the first 8 frames were given. Our model achieves realistic behaviour, outperforms SQAIR, and approaches the quality of the supervised baseline, despite being fully unsupervised. For full effect, the reader is encouraged to watch the animated version in our repository `github.com/jlko/STOVE`. (Best viewed in color.)

objects, requiring $T \cdot O$ sequential recurrence steps in total. This increases the variance of the gradient estimate, slows down training, and hampers scalability. Inspired by Becker-Ehmck et al. (2019), we avoid this cost by *reusing* the dynamics model for the variational distribution. First, we construct the variational distribution $q(z^o_{t,\text{pos}} \mid z^o_{t-1})$ by slightly adjusting the dynamics prediction $p(z^o_{t,\text{pos}} \mid z^o_{t-1})$, using the same mean values but separately predicted standard deviations. Together with an estimate for the *same* object by the object detection network $q(z^o_{t,\text{pos}} \mid x_t)$, we construct a joint estimate by multiplying the two Gaussians and renormalizing, yielding another Gaussian:

$$q(z^o_{t,\text{pos}} \mid z_{t-1}, x_t) \propto q(z^o_{t,\text{pos}} \mid z_{t-1}) \cdot q(z^o_{t,\text{pos}} \mid x_t).$$

Intuitively, this results in a distribution which reconciles the two proposals. A double recurrence is avoided since $q(z_t \mid x_t)$ does not depend on previous timesteps and may thus be computed in parallel for all frames. Similarly, $q(z_t \mid z_{t-1})$ may be computed in parallel for all objects, leading to only $T + O$ sequential recurrence steps total. An additional benefit of this approach is that the information learned by the dynamics network is reused for inference — if $q(z_t \mid x_t, z_{t-1})$ were just another neural network, it would have to essentially relearn the environment's dynamics from scratch, resulting in a waste of parameters and training time. A further consequence is that the image likelihood $p(x_t \mid z_t)$ is backpropagated through the dynamics model, which has been shown to be beneficial for efficient training (Karl et al., 2017; Becker-Ehmck et al., 2019). The same procedure is applied to reconcile velocity estimates from the two networks, where for the image model, velocities $z^o_{t,\text{velo}}$ are estimated from position differences between two consecutive timesteps. The object scales $z^o_{t,\text{scale}}$ are inferred solely from the image model and the latent states $z^o_{t,\text{latent}}$ are given directly by the dynamics network. This then gives us the inference procedure for the full latent state $z^o_t = (z^o_{t,\text{pos}}, z^o_{t,\text{size}}, z^o_{t,\text{velo}}, z^o_{t,\text{latent}})$, as illustrated in Fig. 1 (right).

Despite its benefits, this technique has thus far only been used in environments with a single object or with known state information. A challenge when applying it in a multi-object video setting is to match up the proposals of the two networks. Since the object detection RNN outputs proposals for object locations in an indeterminate order, it is not immediately clear how to find the corresponding proposals from the dynamics network. We have, however, found that a simple matching procedure results in good performance: For each z_t, we assign the object order that results in the minimal difference of $||z_{t,\text{pos}} - z_{t-1,\text{pos}}||$, where $||\cdot||$ is the Euclidean norm. The resulting Euclidean bipartite matching problem can be solved in cubic time using the classic Hungarian algorithm (Kuhn, 1955).

2.4 Conditioning on Actions

In reinforcement learning, an agent interacts with the environment sequentially through actions a_t to optimize a cumulative reward r. To extend STOVE to operate in this setting, we make two changes, yielding a distribution $p(z_t, r_t \mid z_{t-1}, a_{t-1})$.

First, we condition the dynamics model on actions a_t, enabling a conditional prediction based on both state and action. To keep the model invariant to the order of the input objects, the action information is concatenated to each object state z^o_{t-1} before they are fed into the dynamics model. The model has to learn on its own which of the objects in the scene are influenced by the action. To facilitate this, we have found it helpful to also concatenate appearance information from the extracted object patches to the object state. While this patch-wise code could, in general, be obtained using some neural feature extractor, we achieved satisfactory performance by simply using the mean values per color channel.

The second change to the model is the addition of reward prediction. In many RL environments, rewards depend on the interactions between objects. Therefore, the dynamics prediction architecture, presented in Eq. 1, is well suited to also predict rewards. We choose to share the same encoding of object interactions between reward and dynamics prediction and simply apply two different output networks (f in Eq. 1) to obtain the dynamics and reward predictions. The total model is again optimized using the ELBO, this time including the reward likelihood $p(r_t \mid x_{t-1}, z_{t-1})$.

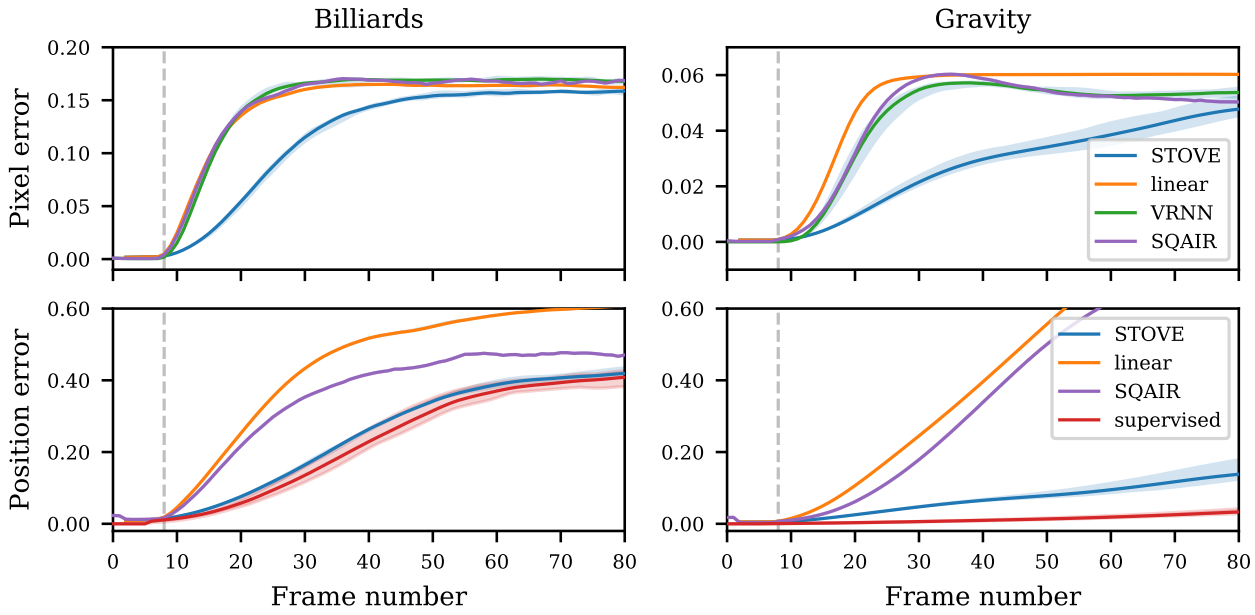


Figure 3: Mean test set performance of our model compared to baselines. Our approach (STOVE) clearly outperforms all unsupervised baselines and is almost indistinguishable from the supervised dynamics model on the billiards task. (Top) Mean squared errors over all pixels in the video prediction setting (the lower, the better). (Bottom) Mean Euclidean distances between predicted and true positions (the lower, the better). All position and pixel values are in $[0, 1]$. In all experiments, the first eight frames are given, all remaining frames are then conditionally generated. The shading indicates the max and min values over multiple training runs with identical hyperparameters. (Best viewed in color.)

3 Experimental Evidence

In order to evaluate our model, we compare it to baselines in three different settings: First, pure video prediction, where the goal is to predict future frames of a video given previous ones. Second, the prediction of future object positions, which may be relevant for downstream tasks. Third, we extend one of the video datasets to a reinforcement learning task and investigate how our physics model may be utilized for sample-efficient, model-based reinforcement learning. With this paper, we also release a PyTorch implementation of STOVE. [2]

3.1 Video and State Modeling

Inspired by (Watters et al., 2017), we considered grayscale videos of objects moving according to physical laws. In particular, we opted for the commonly used bouncing billiards balls dataset, as well as a dataset of gravitationally interacting balls. For further details on the datasets, see Appendix C. When trained using a single GTX 1080 Ti, STOVE converges after about 20 hours on this data. As baselines, we compared to VRNNs (Chung et al., 2015) and SQAIR (Kosiorek et al., 2018). To allow for a fair comparison, we fixed the number of objects predicted by SQAIR to the correct amount. Furthermore, we compared to a supervised baseline: Here, we considered the ground truth positions and velocities to be fully observed, and trained our dynamics model on them, resembling the setting of Battaglia et al. (2016). Since our model needs to infer object states from pixels, this baseline provides an upper bound on the predictive performance we can hope to achieve with our model. In turn, the size of the performance gap between the two is a good indicator of the quality of our state-space model. As an ablation, we also report the results obtained by combining our image model with a simple linear physics model, which linearly extrapolates the objects' trajectories. Since VRNN does not reason about object positions, we only evaluated it on the video prediction task. Similarly, the supervised baseline does not reason about images and was only considered for the position prediction task. For more information on the baselines, see Appendix D.

[2] The code can be found in the GitHub repository `https://github.com/jlko/STOVE`. It also contains animated versions of the videos predicted by our model and the baselines.

Table 1: Predictive performance of our approach, the baselines, and ablations (the lower, the better, best unsupervised values are bold). STOVE outperforms all unsupervised baselines and is almost indistinguishable from the supervised model on the billiards task. The values are computed by summing the prediction errors presented in Fig. 3 in the time interval $t \in [9, 18]$, i.e., the first ten predicted timesteps. In parentheses, standard deviations across multiple training runs are given.

	STOVE (ours)	VRNN	SQAIR	Linear	Supervised
Billiards (pixels)	$\mathbf{0.240 \pm 0.014}$	0.526 ± 0.014	0.591	0.844 ± 0.003	–
Billiards (positions)	$\mathbf{0.418 \pm 0.020}$	–	0.804	1.348 ± 0.014	0.232 ± 0.037
Gravity (pixels)	$\mathbf{0.040 \pm 0.003}$	0.055 ± 0.012	0.070	0.196 ± 0.002	–
Gravity (positions)	$\mathbf{0.142 \pm 0.007}$	–	0.194	0.493 ± 0.004	0.014 ± 0.003

Fig. 2 illustrates predictions on future object positions made by the models, after each of them was given eight consecutive frames from the datasets. Visually, the predictions produced by STOVE are on par with the supervised baseline, as well as with other supervised approaches from the literature such as (Watters et al., 2017), which assume access to precise ground truth states at training time. Despite the fact that our dynamics model is only conditioned on the previous timestep, STOVE is able to generate realistic looking rollouts over hundreds of timesteps. This is in contrast to previous work such as by Watters et al. (2017), who explicitly condition on multiple previous timesteps and include auxiliary loss terms as opposed to simply maximizing the ELBO.

Fig. 3 depicts the reconstruction and prediction errors of the various models: Each model is given eight frames of video from the test set as input, which it then reconstructs. Conditioned on this input, the models predict the object positions or resulting video frames for the following 72 timesteps. The predictions are evaluated on ground truth data by computing the mean squared error between pixels, and the Euclidean distance between positions based on the best available object matching. We outperform all baselines on both the state and the image prediction task by a large margin. Additionally, we perform strikingly close to the supervised model. Table 1 underlines these results with concrete numbers. For the gravitational data, the prediction task appears easier, as both our model as well as the supervised baseline achieve lower errors than in the billiards task. However, in this regime of easy prediction, precise access to the object states becomes more important, which is likely the reason why the gap between our approach and the supervised baseline is slightly more pronounced. Despite this, STOVE produces high-quality rollouts and outperforms the unsupervised baselines.

3.2 Model-Based Control

To explore the usefulness of STOVE for reinforcement learning, we extend the billiards dataset into a reinforcement learning task. Now, the agent controls one of the balls using nine actions, which correspond to moving in one of the eight (inter)cardinal directions and staying at rest. The goal is to avoid collisions with the other balls, which elastically bounce off of each other, the walls, and the controlled ball. A negative reward of -1 is given whenever the controlled ball collides with one of the others. Starting with a random policy, we iteratively gather observations from the environment, i. e. sequences of images, actions, and rewards. Using these, we train our model as described in Sec. 2.4. To obtain a policy based on our world model, we use Monte-Carlo tree search (MCTS), leveraging our model as a simulator for planning. Using this policy, we gather more observations and apply them to refine the world model. As an upper bound on the performance achievable in this manner, we report the results obtained by MCTS when the real environment is used for planning. As a model-free baseline, we consider PPO (Schulman et al., 2017), which is a state-of-the-art algorithm on comparable domains such as Atari games. To explore the effect of the availability of state information, we also run PPO on a version of the environment in which, instead of images, the ground-truth object positions and velocities are observed directly.

Learning curves for each of the agents are given in Fig. 4 (left), reported at intervals of $10\,000$ samples taken from the environment, up to a total of $130\,000$. For our model, we collect the first $50\,000$ samples using a random policy, to provide an initial training set. After that, the described training loop is used, iterating between collecting $10\,000$ observations using an MCTS-based policy and refining the model using examples sampled from the pool of previously seen observations. After $130\,000$ samples, PPO has not yet seen enough samples to converge, whereas our model quickly learns to meaningfully model the environment and thus produces a better policy at this stage. Even when PPO is trained on ground truth states, MCTS based on STOVE remains comparable.

After training each model to convergence, the final performance of all approaches is reported in Fig. 4 (right). In this case, PPO achieves slightly better results, however it only converges after training for approximately $5\,000\,000$ steps, while our approach only uses $130\,000$ samples. After around $3\,000\,000$ steps, PPO does eventually surpass the performance of STOVE-based MCTS. Additionally, we find that MCTS on STOVE yields almost the same performance as on the real environment, indicating that it can be used to anticipate and avoid collisions accurately.

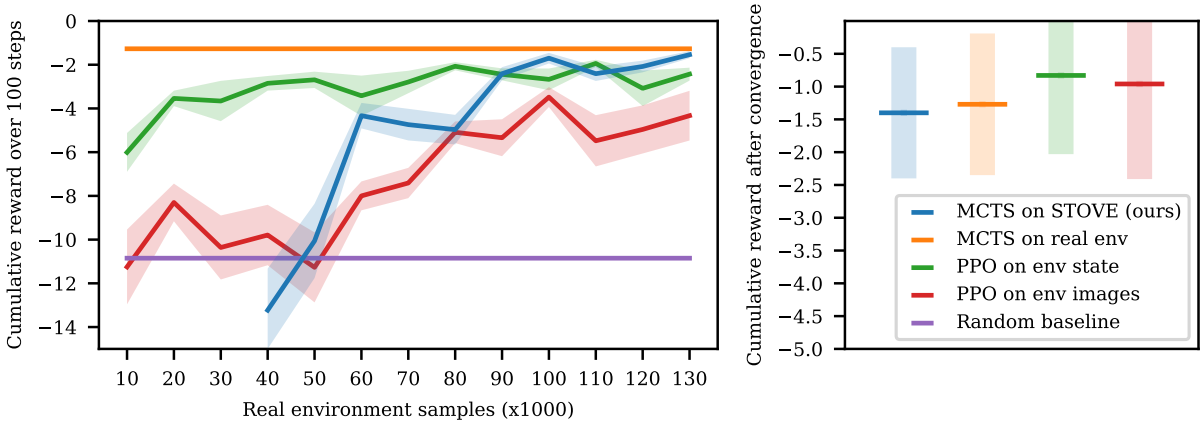


Figure 4: Comparison of all models on sample efficiency and final performance. (Left) Mean cumulative reward over 100 steps on the environment, averaged over 100 environments, using the specified policy. The shaded regions correspond to one-tenth of a standard deviation. In addition to the training curves, two constant baselines are shown, one representing a random policy and one corresponding to the MCTS based policy when using the real environment as a simulator. (Right) Final performance of all approaches, after training each model to convergence. The shaded region corresponds to one standard deviation. (Best viewed in color.)

4 Related Work

Multiple lines of work with the goal of video modeling or prediction have emerged recently. Prominently, the supervised modeling of physical interactions from videos has been investigated by Fragkiadaki et al. (2015), who train a model to play billiards with a single ball. Similarly, graph neural networks have been trained in a supervised fashion to predict the dynamics of objects from images (Watters et al., 2017; Sanchez-Gonzalez et al., 2018). Janner et al. (2019) show successful planning based on learned interactions, but assume access to image segmentations. Several unsupervised approaches address the problem by fitting the parameters of a physics engine to data (Jaques et al., 2019; Wu et al., 2016, 2015). This necessitates specifying in advance which physical laws govern the observed interactions. In the fully unsupervised setting, mainly unstructured variational approaches have been explored (Babaeizadeh et al., 2017; Chung et al., 2015; Krishnan et al., 2015). However, without the explicit notion of objects, their performance in scenarios with interacting objects remains limited. Nevertheless, unstructured video models have recently been applied to model-based RL and have been shown to improve sample efficiency when used as a simulator for the real environment (Oh et al., 2015; Kaiser et al., 2019).

Only a small number of works incorporate objects into unsupervised video models. Notable examples are SQAIR (Kosiorek et al., 2018), R-NEM (Van Steenkiste et al., 2018), DDPAE (Hsieh et al., 2018), and COBRA (Watters et al., 2019). R-NEM learns a mixture model via expectation-maximization unrolled through time and handles interactions between objects in a factorized fashion. However, it lacks an explicitly structured latent space, requires noise in the input data to avoid local minima, and struggles with the constancy of object appearances ("wobbling"), albeit less pronounced than VRNNs. Both DDPAE and SQAIR extend the AIR approach to work on videos using standard recurrent architectures. As discussed, this introduces a double recurrence over objects and time, which is detrimental for performance. However, SQAIR is capable of handling a varying number of objects, which is not something we consider in this paper. Finally, COBRA presents a model-based RL approach based on MONet, but is restricted to environments with non-interacting objects and only uses one-step search to build its policy.

5 Conclusion

We introduced STOVE, a structured, object-aware model for unsupervised video modeling and planning. It combines recent advances in unsupervised image modeling and physics prediction into a single compositional state-space model. The resulting joint model explicitly reasons about object positions and velocities, and is capable of generating highly accurate video predictions in domains featuring complicated non-linear interactions between objects. As our experimental evaluation shows, it outperforms previous unsupervised approaches and even approaches the performance and visual quality of a supervised model.

Additionally, we presented an extension of the video learning framework to the RL setting. Our experiments demonstrate that our model may be utilized for sample-efficient model-based control in a visual domain, making headway towards a long standing goal of the model-based RL community. In particular, STOVE yields good performance with more than one order of magnitude fewer samples compared to the model-free baseline, even when paired with a relatively simple planning algorithm like MCTS.

At the same time, STOVE also makes several assumptions for the sake of simplicity. Relaxing them provides interesting avenues for future research. First, we assume a fixed number of objects, which may be avoided by performing dynamic object propagation and discovery like in SQAIR. Second, we have inherited the assumption of rectangular object masks from AIR. Applying a more flexible model such as MONet (Burgess et al., 2019) or GENESIS (Engelcke et al., 2019) may alleviate this, but also poses additional challenges, especially regarding the explicit modeling of movement. Finally, the availability of high-quality learned state space models enables to the use of more sophisticated planning algorithms in visual domains (Chua et al., 2018). In particular, by combining planning with policy and value networks, model-free and model-based RL may be integrated into a comprehensive system (Buckman et al., 2018).

Acknowledgments. The authors thank Adam Kosiorek for his assistance with the SQAIR experiments. KK acknowledges the support of the Rhine-Main universities' network for "Deep Continuous-Discrete Machine Learning" (DeCoDeML).

References

M. Babaeizadeh, C. Finn, D. Erhan, R. H. Campbell, and S. Levine. Stochastic variational video prediction. In *Proceedings of ICLR*, 2017.

P. Battaglia, R. Pascanu, M. Lai, D. J. Rezende, et al. Interaction networks for learning about objects, relations and physics. In *Proceedings of NeurIPS*, pages 4502–4510, 2016.

P. W. Battaglia, J. B. Hamrick, V. Bapst, A. Sanchez-Gonzalez, V. Zambaldi, M. Malinowski, A. Tacchetti, D. Raposo, A. Santoro, R. Faulkner, et al. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*, 2018.

P. Becker-Ehmck, J. Peters, and P. Van Der Smagt. Switching linear dynamics for variational bayes filtering. In *Proceedings of ICML*, 2019.

J. Buckman, D. Hafner, G. Tucker, E. Brevdo, and H. Lee. Sample-efficient reinforcement learning with stochastic ensemble value expansion. In *Proceedings of NeurIPS*, pages 8224–8234, 2018.

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

K. Chua, R. Calandra, R. McAllister, and S. Levine. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. In *Proceedings of NeurIPS*, pages 4754–4765, 2018.

J. Chung, K. Kastner, L. Dinh, K. Goel, A. C. Courville, and Y. Bengio. A recurrent latent variable model for sequential data. In *Proceedings of NeurIPS*, pages 2980–2988, 2015.

R. Coulom. Efficient selectivity and backup operators in monte-carlo tree search. In *Computers and Games*, pages 72–83. Springer Berlin Heidelberg, 2007.

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

S. A. Eslami, N. Heess, T. Weber, Y. Tassa, D. Szepesvari, G. E. Hinton, et al. Attend, infer, repeat: Fast scene understanding with generative models. In *Proceedings of NeurIPS*, pages 3225–3233, 2016.

K. Fragkiadaki, P. Agrawal, S. Levine, and J. Malik. Learning visual predictive models of physics for playing billiards. *arXiv preprint arXiv:1511.07404*, 2015.

I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Proceedings of NeurIPS*, pages 2672–2680, 2014.

U. Grenander. *Lectures in Pattern Theory: Vol. 2 Pattern Analysis*. Springer-Verlag, 1976.

S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

J.-T. Hsieh, B. Liu, D.-A. Huang, L. F. Fei-Fei, and J. C. Niebles. Learning to decompose and disentangle representations for video prediction. In *Proceedings of NeurIPS*, pages 517–526, 2018.

M. Janner, S. Levine, W. T. Freeman, J. B. Tenenbaum, C. Finn, and J. Wu. Reasoning about physical interactions with object-centric models. In *Proceedings of ICLR*, 2019.

M. Jaques, M. Burke, and T. Hospedales. Physics-as-inverse-graphics: Joint unsupervised learning of objects and physics from video. *arXiv preprint arXiv:1905.11169*, 2019.

L. Kaiser, M. Babaeizadeh, P. Milos, B. Osinski, R. H. Campbell, K. Czechowski, D. Erhan, C. Finn, P. Kozakowski, S. Levine, et al. Model-based reinforcement learning for atari. *arXiv preprint arXiv:1903.00374*, 2019.

M. Karl, M. Soelch, J. Bayer, and P. van der Smagt. Deep Variational Bayes Filters: Unsupervised Learning of State Space Models from Raw Data. In *Proceedings of ICLR*, 2017.

D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *Proceedings of ICLR*, 2015.

D. P. Kingma and M. Welling. Auto-encoding variational bayes. In *Proceedings of ICLR*, 2014.

A. Kosiorek, H. Kim, Y. W. Teh, and I. Posner. Sequential attend, infer, repeat: Generative modelling of moving objects. In *Proceedings of NeurIPS*, pages 8606–8616, 2018.

R. G. Krishnan, U. Shalit, and D. Sontag. Deep kalman filters. *arXiv preprint arXiv:1812.08434*, 2015.

H. W. Kuhn. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97, 1955.

J. Oh, X. Guo, H. Lee, R. L. Lewis, and S. Singh. Action-conditional video prediction using deep networks in atari games. In *Proceedings of NeurIPS*, pages 2845–2853, 2015.

A. Sanchez-Gonzalez, N. Heess, J. T. Springenberg, J. Merel, M. Riedmiller, R. Hadsell, and P. Battaglia. Graph networks as learnable physics engines for inference and control. In *Proceedings of ICML*, 2018.

A. Santoro, D. Raposo, D. G. Barrett, M. Malinowski, R. Pascanu, P. Battaglia, and T. Lillicrap. A simple neural network module for relational reasoning. In *Proceedings of NeurIPS*, pages 4967–4976, 2017.

J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

K. Stelzner, R. Peharz, and K. Kersting. Faster attend-infer-repeat with tractable probabilistic models. In *Proceedings of ICML*, pages 5966–5975, 2019.

R. S. Sutton and A. G. Barto. *Reinforcement learning: An introduction*. MIT Press, Cambridge, 2011.

S. Van Steenkiste, M. Chang, K. Greff, and J. Schmidhuber. Relational neural expectation maximization: Unsupervised discovery of objects and their interactions. In *Proceedings of ICLR*, 2018.

A. Vergari, R. Peharz, N. Di Mauro, A. Molina, K. Kersting, and F. Esposito. Sum-product autoencoding: Encoding and decoding representations using sum-product networks. In *Proceedings of AAAI*, 2018.

N. Watters, D. Zoran, T. Weber, P. Battaglia, R. Pascanu, and A. Tacchetti. Visual interaction networks: Learning a physics simulator from video. In *Proceedings of NeurIPS*, pages 4539–4547, 2017.

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

J. Wu, I. Yildirim, J. J. Lim, B. Freeman, and J. Tenenbaum. Galileo: Perceiving physical object properties by integrating a physics engine with deep learning. In *Proceedings of NeurIPS*, pages 127–135, 2015.

J. Wu, J. J. Lim, H. Zhang, J. B. Tenenbaum, and W. T. Freeman. Physics 101: Learning physical object properties from unlabeled videos. In *Proceedings of BMVC*, 2016.

C. Zhang, J. Butepage, H. Kjellstrom, and S. Mandt. Advances in variational inference. *IEEE transactions on pattern analysis and machine intelligence*, 2018.

J. Zhou, G. Cui, Z. Zhang, C. Yang, Z. Liu, and M. Sun. Graph neural networks: A review of methods and applications. *arXiv preprint arXiv:1812.08434*, 2018.

A Reconstructions: Sprites Data

SuPAIR does not need a latent description of the objects' appearances. Nevertheless, object reconstructions can be obtained by using a variant of approximate MPE (most probable explanation) in the sum-product networks as proposed by Vergari et al. (2018). We follow the AIR approach and reconstruct each object separately and paste it into the canvas using spatial transformers. Unlike AIR, SuPAIR explicitly models the background using a separate background SPN. A reconstruction of the background is also obtained using MPE.

To demonstrate the capabilities of our image model, we also trained our model on a variant of the gravity data in which the round balls were replaced by a random selection of four different sprites of the same size. Fig. 5 shows the reconstructions obtained from SuPAIR when trained on these more complex object shapes.

B Model Details

Here, we present additional details on the architecture and hyperparameters of STOVE.

B.1 Inference Architecture

The object detection network for $q(z_{t,\text{where}} \mid x_t)$ is realised by an LSTM (Hochreiter and Schmidhuber, 1997) with 256 hidden units, which outputs the mean and standard deviation of the objects' two-dimensional position and size distributions, i.e. $q(z^o_{t,\text{pos, size}} \mid x_t)$ with $2 \cdot 2 \cdot 2 = 8$ parameters per object. Given such position distributions for two consecutive timesteps $q(z_{t-1,\text{pos}} \mid x_{t-1}), q(z_{t,\text{pos}} \mid x_t)$, with parameters $\mu_{z^o_{t-1,\text{pos}}}, \sigma_{z^o_{t-1,\text{pos}}}, \mu_{z^o_{t,\text{pos}}}, \sigma_{z^o_{t,\text{pos}}}$, the following velocity estimate based on the difference in position is constructed:

$$q(z^o_{t,\text{velo}} \mid x_t, x_{t-1}) = \mathcal{N}(\mu_{z^o_{t,\text{pos}}} - \mu_{z^o_{t-1,\text{pos}}}, \sigma^2_{z^o_{t,\text{pos}}} + \sigma^2_{z^o_{t-1,\text{pos}}}).$$

As described in Sec. 2.3, positions and velocities are also inferred from the dynamics model as $q(z^o_{t,\text{pos}} \mid z_{t-1})$ and $q(z^o_{t,\text{velo}} \mid z_{t-1})$. A joint estimate, including information from both image model and dynamics prediction, is obtained by multiplying the respective distributions and renormalizing. Since both q-distributions are Gaussian, the normalized product is again Gaussian, where mean and standard deviation are given by

$$\begin{aligned}
q(z_t \mid x_t, z_{t-1}) &\propto q(z_t \mid x_t) \cdot q(z_t \mid z_{t-1}) \\
&= \mathcal{N}(z_t; \mu_{t,i}, \sigma^2_{t,i}) \cdot \mathcal{N}(z_t; \mu_{t,d}, \sigma^2_{t,d}) \\
&= \mathcal{N}(z_t; \mu_t, \sigma^2_t) \\
\mu_t &= \frac{\sigma^2_{t,d}\,\mu_{t,i} + \sigma^2_{t,i}\,\mu_{t,d}}{\sigma^2_{t,d} + \sigma^2_{t,i}} \\
\frac{1}{\sigma^2_t} &= \frac{1}{\sigma^2_{t,d}} + \frac{1}{\sigma^2_{t,i}}\,,
\end{aligned}$$

where we relax our notation for readability $z_t \in [z^o_{t,\text{pos}}, z^o_{t,\text{velo}}]$ and the indices i and d refer to the parameters obtained from the image and dynamics model. This procedure is applied independently for the positions and velocities of each object.

For $z^o_{t,\text{latent}}$, we choose dimension 12, such that a full state $z^o_t = (z^o_{t,\text{pos}}, z^o_{t,\text{size}}, z^o_{t,\text{velo}}, z^o_{t,\text{latent}})$ is 18-dimensional.

B.2 Graph Neural Network

The dynamics prediction is given by the following series of transformations applied to each input state of shape $(\texttt{batch size}, \texttt{number of objects}, l)$, where $l = 16$, since currently, size information is not propagated through the dynamics prediction.

- S_1: Encode input state with linear layer $[l, 2l]$.
- S_2: Apply linear layer $[2l, 2l]$ to S_1 followed by ReLU non-linearity.
- S_3: Apply linear layer $[2l, 2l]$ to S_2 and add result to S_2. This gives the dynamics prediction without relational effects, corresponding to $g(z^o_t)$ in Eq. 1.

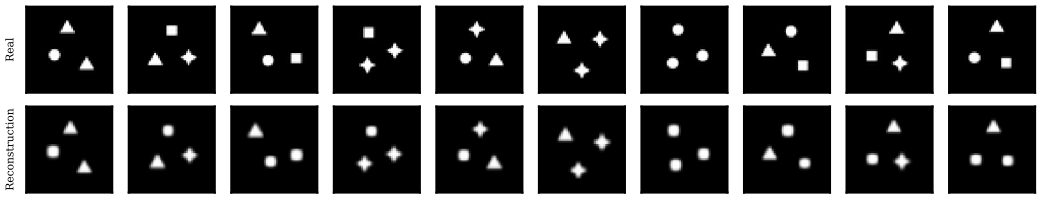


Figure 5: Reconstructions obtained from our image model when using more varied shapes.

- C_1: The following steps obtain the relational aspects of dynamics prediction, corresponding to $h(z_t^o, z_t^{o'})$ in Eq. 1. Concatenate the encoded state S_1^o pairwise with all state encoding, yielding a tensor of shape $(\texttt{batch size}, \texttt{number of objects}, \texttt{number of objects}, 4l)$.
- C_2: Apply linear layer $[4l, 4l]$ to C_1 followed by ReLU.
- C_3: Apply linear layer $[4l, 2l]$ to C_2 followed by ReLU.
- C_4: Apply linear layer $[2l, 2l]$ to C_3 and add to C_3.
- A_1: To obtain attention coefficients $\alpha(z_t^o, z_t^{o'})$, apply linear layer $[4l, 4l]$ to C_1 followed by ReLU.
- A_2: Apply linear layer $[4l, 2l]$ to A_1 followed by ReLU.
- A_3: Apply linear layer $[2l, 1]$ to A_2 and apply exponential function.
- R_1: Multiply C_4 with A_3, where diagonal elements of A_3 are masked out to ensure that R_1 only covers cases where $o \neq o'$.
- R_2: Sum over R_1 for all o', to obtain tensor of shape $(\texttt{batch size}, \texttt{number of objects}, 2l)$. This is the relational dynamics prediction.
- D_1: Sum relational dynamics R_2 and self-dynamics S_3, obtaining the input to f in Eq. 1.
- D_2: Apply linear layer $[2l, 2l]$ to D_1 followed by tanh non-linearity.
- D_3: Apply linear layer $[2l, 2l]$ to D_2 followed by tanh non-linearity and add result to D_2.
- D_4: Concatenate D_3 and S_1, and apply linear layer $[4l, 2l]$ followed by tanh.
- D_5: Apply linear layer $[2l, 2l]$ to D_4 and add result to D_4 to obtain final dynamics prediction.

The output D_5 has shape $(\texttt{batch size}, \texttt{number of objects}, 2l)$, twice the size of means and standard deviations over the next predicted state.

For the model-based control scenario, the one-hot encoded actions ($\texttt{batch size}$, $\texttt{action space}$) are transformed with a linear layer [$\texttt{action space}$, $\texttt{number of objects} \cdot \texttt{encoding size}$] and reshaped to ($\texttt{action space}$, $\texttt{number of objects}$, $\texttt{encoding size}$). The action embedding and the object appearances ($\texttt{batch size}$, $\texttt{number of objects}$, 3) are then concatenated to the input state. The rest of the dynamics prediction follows as above. The reward prediction consists of the following steps:

- H_1: Apply linear layer $[2l, 2l]$ to D_1 followed by ReLU.
- H_2: Apply linear layer $[2l, 2l]$ to H_1.
- H_3: Sum over object dimension to obtain tensor of shape $(\texttt{batch size}, l)$.
- H_4: Apply linear layer $[l, l/2]$ to H_3 followed by ReLU.
- H_5: Apply linear layer $[l/2, l/4]$ to H_4 followed by ReLU.
- H_5: Apply linear layer $[l/4, l]$ to H_4 followed by a sigmoid non-linearity.

H_5 then gives the final reward prediction.

B.3 State Initialization

In the first two timesteps, we cannot yet apply STOVE's main inference step $q(z_t \mid z_{t-1}, x_t, x_{t-1})$ as described above. In order to initialize the latent state over the first two frames, we apply a simplified architecture and only use a partial state at $t = 0$.

At $t = 0$, $z_0 \sim q(z_{0,(\text{pos, size})} \mid x_0)$ is given purely by the object detection network, since no previous states, which could be propagated, exist. z_0 is incomplete insofar as it does not contain velocity information or latents. At $t = 1$, $q(z_{1,\text{pos, size}} \mid x_1, x_0)$ is still given purely based on the object detection network. Note that for a dynamics prediction of z_1, velocity information at $t = 0$ would need to be available. However, at $t = 1$, velocities can be constructed based on the differences between the previously inferred object positions. We sample $z_{1,\text{latent}}$ from the prior Gaussian distribution to assemble the first full initial state z_1. At $t \geq 2$, the full inference network can be run: States are inferred both from the object detection network $q(z_t \mid x_t, x_{t-1})$ as well as propagated using the dynamics model $q(z_t \mid z_{t-1})$.

In the generative model, similar adjustments are made: $p(z_{0,\text{pos, size}})$ is given by a uniform prior, velocities and latents are omitted. At $t = 1$, velocities are sampled from a uniform distribution in planar coordinates $p(z_{1,\text{velo}})$ and positions are given by a simple linear dynamics model $p(z_{1,\text{pos}} \mid z_{0,\text{pos}}, z_{1,\text{velo}}) = \mathcal{N}(z_{0,\text{pos}} + z_{1,\text{velo}}, \sigma)$. Latents $z_{1,\text{latent}}$ are sampled from a Gaussian prior. Starting at $t = 2$, the full dynamics model is used.

B.4 Training Procedure

Our model was trained using the Adam optimizer (Kingma and Ba, 2015), with a learning rate of $2 \times 10^{-3} \exp(-40 \times 10^{-3} \cdot \text{step})$ for a total of $83\,000$ steps with a batch size of 256.

C Data Details

For the billiards and gravitational data, 1000 sequences of length 100 were generated for training. From these, subsequences of lengths 8 were sampled and used to optimize the ELBO. A test dataset of 300 sequences of length 100 was also generated and used for all evaluations. The pixel resolution of the dataset was 32×32 for the billiards data and 50×50 for the gravity data. All models for video prediction were learned on grayscale data. The $O = 3$ balls were initialised with uniformly random positions and velocities, rejecting configurations with overlap. They are rendered using anti-aliasing. The billiards data models the balls as circular objects, which perform elastic collision with each other or the walls of the environment. For the gravity data, the balls are modeled as point masses, where, following Watters et al. (2017), we clip the gravitational force to avoid slingshot effects. Also, we add an additional basin of attraction towards the center of the canvas and model the balls in their center off mass system to avoid drift. Velocities here are initialised orthogonal to the center of the canvas for a stabilising effect. For full details we refer to the file `envs.py` in the provided code.

D Baselines for Video Modeling

Following Kosiorek et al. (2018), we experimented with different hyperparameter configurations for VRNNs. We varied the sizes of the hidden and latent states $[h, z]$, experimenting with the values $[256, 16]$, $[512, 32]$, $[1024, 64]$, and $[2048, 32]$. We found that increasing the model capacity beyond $[512, 32]$ did not yield large increases in performance, which is why we chose the configuration $[512, 32]$ for our experiments. Our VRNN implementation is written in PyTorch and based on `https://github.com/emited/VariationalRecurrentNeuralNetwork`.

SQAIR can handle a variable number of objects in each sequence. However, to allow for a fairer comparison to STOVE, we fixed the number of objects to the correct number. This means that in the first timestep, exactly three objects are discovered, which are then propagated in all following timesteps, without further discoveries. Our implementation is based on the original implementation provided by the authors at `https://github.com/akosiorek/sqair`.

The linear baseline was obtained as follows: For the first 8 frames, we infer the full model state using STOVE. We then take the last inferred positions and velocities of each object and predict future positions by assuming constant, uniform motions for each object. We do not allow objects to leave the frame, i. e. when objects reach the canvas boundary after some timesteps, they stick to it.

Since our dynamics model requires only object positions and velocities as input, it is trivial to construct a supervised baseline for our physics prediction by replacing the SuPAIR-inferred states with real, ground-truth states. On these, the model can then be trained in supervised fashion.

E Details on the Reinforcement Learning Models

Our MCTS implementation uses the standard UCT formulation for exploration/exploitation. The c parameter is set to 1. in all our experiments. Since the environment does not provide a natural endpoint, we cut off all rollouts at a depth of 20 timesteps. We found this to be a good trade-off between runtime and accuracy.

When expanding a node on the true environment, we compute the result of the most promising action, and then start a rollout using a random policy from the resulting state. For the final evaluation, a total of 200 nodes are expanded. To better utilize the GPU, a slightly different approach is used for STOVE. When we expand a node in this setting, we predict the results of all actions simultaneously, and compute a rollout from each resulting position. In turn, only 50 nodes are expanded. To estimate the node value function, the average reward over all rollouts is propagated back to the root and each node's visit counter is increased by 1. Furthermore, we discount the reward predicted STOVE with a factor of 0.95 per timestep to account for the higher uncertainty of longer rollouts. This is not done in the baseline running on the real environment, since it behaves deterministically.

For PPO, we employ a standard convolutional neural network as an actor-critic for the evaluation on images and a MLP for the evaluation on states. The image network consists of two convolutional layers, each using 32 output filters with a kernel size of 4 and 3 respectively and a stride of 2. The MLP consists of two fully connected layers with 128 and 64 hidden units. In both cases, an additional fully connected layer links the outputs of the respective base to an actor and a critic head. For the convolutional base, this linking layer employs 512 hidden units, for the MLP 64. All previously mentioned layers use rectified linear activations. The actor head predicts a probability distribution over next actions using a softmax activation function while the critic head outputs a value estimation for the current state using a linear prediction. We tested several hyperparameter configurations but found the following to be the most efficient one. To update the actor-critic architecture, we sample 32 trajectories of length 16 from separate environments in every batch. The training uses an Adam optimizer with a learning rate of 2×10^{-4} and and ϵ value of 1×10^{-5}. The clipping parameter of PPO is set to 1×10^{-1}. We update the network for 4 epochs in each batch using 32 mini-batches of the sampled data. The value loss is weighted at 5×10^{-1} and the entropy coefficient is set to 1×10^{-2}.