STRUCTURED OBJECT-AWARE PHYSICS PREDICTION FOR VIDEO MODELING AND PLANNING

Anonymous authors
Paper under double-blind review

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., 2017).

Moreover, inspired by human vision, understanding images as compositions of objects and background has recently proven to be a successful prior for building unsupervised models of single images. The structured nature of approaches such as AIR (Eslami et al., 2016), MONet (Burgess et al., 2019), and SuPAIR (Stelzner et al., 2019) provides two key advantages over unstructured image models such as variational autoencoders (Kingma & 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.

Thus, compositional modeling 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. Existing works in this area, such as SQAIR (Kosiorek et al., 2018), DDPAE (Hsieh et al., 2018), R-NEM (Van Steenkiste et al., 2018), 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 for physical interactions, we propose STOVE, a structured object-aware video prediction 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 variational inference in this model, we provide a novel inference architecture, which reuses the learned generative physics model in the recognition network. As we will demonstrate, our model generates realistic 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 & Barto, 2011). However, recent work in the area of model-based RL has shown how powerful video prediction networks can be employed to improve sample efficiency when learning to play Atari videogames (Oh et al., 2015; Kaiser et al., 2019). 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.

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

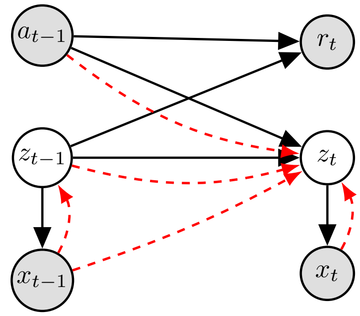

Figure 1: Depiction of the graphical model of our approach. Black arrows denote the generative mechanism and red arrows the inference procedure. The variational distribution $q(z_t \mid z_{t-1}, x_t)$ is formed by combining predictions from the dynamics model $p(z_t \mid z_{t-1})$ and the object detection network $q(z_t \mid x_t)$. For the RL domain, our approach is extended by action conditioning and reward prediction.

objects, each of which occupies a rectangular region in the image, specified by positional parameters $z^o_{\text{where}} = (z^o_{\text{pos}}, z^o_{\text{size}})$. The visual content of each object is described by a latent code z^o_{what}. By decoding z^o_{what} 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 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 $p(x_t \mid z_{t_{\text{where}}})$. 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 *order 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., 2018; 2016; 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 which allows the network to focus on specific object pairs. Finally, 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.

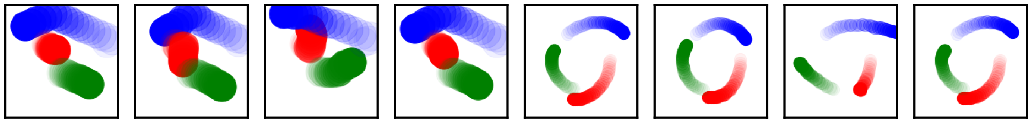

Figure 2: Visualisation of predicted object positions. Each illustrations shows object positions from the real environment, predictions made by our model, SQAIR, and the supervised baseline, after the first 8 frames were given. Our model achieves realistic behaviour, outperforms SQAIR, and approaches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

2.3 Joint State-Space Model

Next, we assemble a single 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 $p(x, z) = p(z_0)p(x_0 \mid z_0) \prod_t p(z_t \mid z_{t-1})p(x_t \mid z_t)$, where we model $p(z_0)$ as simple uniform distributions for the structured variables, and Gaussians for the latent codes $z_{0,\text{latent}}$.

Our model is trained on given video sequences x by maximizing the evidence lower bound (ELBO) $\mathbb{E}_{q(z|x)} [\log p(x, z) - \log q(z \mid x)]$. 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 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 $q(z^o_{t,\text{(pos,vel)}} \mid x_t)$, 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 good performance (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 $z^o_t = (z^o_{t,\text{pos}}, z^o_{t,\text{size}}, z^o_{t,\text{velo}}, z^o_{t,\text{latent}})$.

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 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

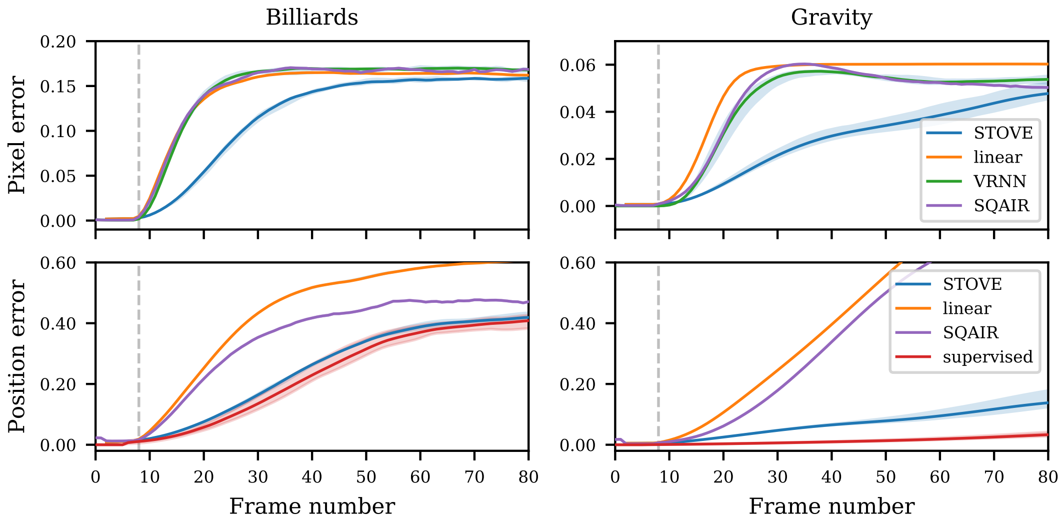

Figure 3: Mean test set performance of our approach, the supervised ablation, and baselines. Our approach (STOVE) clearly outperforms all baselines and is almost indistinguishable from the supervised baseline on the billiards task. (Top) Mean squared errors over all pixels in the video prediction setting (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 the given, all remaining frames are then conditionally generated. The shading indicates the max and min values over ten different training runs with identical hyperparameters. (Best viewed in color.)

results in good performance. That is, for each z_t, we assign the object in the order that results in the minimal difference of $||z_{t,\text{(pos, vel)}} - z_{t-1,\text{(pos, vel)}}||$, where $|| \cdot ||$ is the Euclidean norm.

2.4 Conditioning on Actions

In reinforcement learning (RL), 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 on 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_t^o 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 a reward prediction. In many RL environments, rewards depend on the interactions between objects. Therefore, the graph neural network for predicting dynamics, presented in eq. (1), translates well to a reward prediction model. 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 $p(r_t \mid x_{t-1}, z_{t-1})$.

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,

Table 1: Predictive performance of our approach, the baselines and ablations. Shown are the mean Euclidian distances between predicted and true positions (the lower, the better). Best values are bold. As one can see, our approach outperforms all baselines and is almost indistinguishable from the supervised baseline on the billiards task. The values are computed by averaging the prediction errors presented in fig. 3 in the time interval $t \in [9, 18]$, i.e., the first ten predicted timesteps. In brackets, standard deviations across multiple training runs are given.

	Ours	VRNN	SQAIR	Linear	Supervised
Billiards (pixels)	$\mathbf{0.0240(14)}$	$0.0526(14)$	0.0591	$0.084\,42(28)$	–
Billiards (states)	$\mathbf{0.0418(20)}$	–	0.0804	$0.1348(14)$	$0.0232(37)$
Gravity (pixels)	$\mathbf{0.004\,01(29)}$	$0.0055(12)$	0.0070	$0.019\,56(23)$	–
Gravity (states)	$\mathbf{0.014\,23(72)}$	–	0.0194	$0.049\,32(42)$	$0.001\,42(25)$

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 used for sample-efficient, model-based reinforcement learning. With this paper, we also release a PyTorch implementation of STOVE. [1]

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 A.4. 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 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 frame 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 A.5.

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, our model is able to generate realistic looking rollouts over hundreds of timesteps. This is in contrast to previous work such as 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 by computing the mean squared pixel and mean position errors with respect to the ground truth data. 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 gravitational baseline perform better than in the billiards task. However, in this regime of easy prediction, precise access to the object states becomes more important, which is why the gap between our approach and the supervised baseline is slightly

[1] The code can be found at an anonymized GitHub repository at `https://github.com/ICLR20/STOVE`. This is also where animated versions of the plots and our model and baselines can be found.

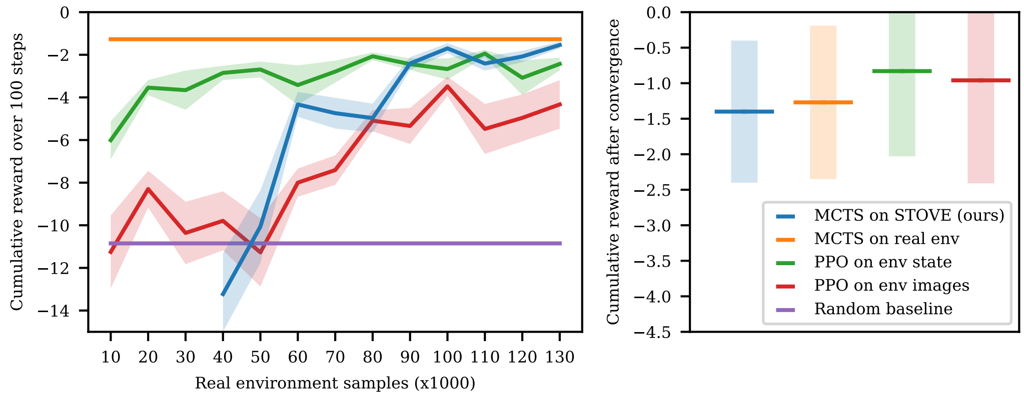

Figure 4: Comparison of all models on sample efficiency and final performance. (Left) The curves represent the mean cumulative reward over 100 steps on the environment, executed over 100 environments, using the specified policy. The shaded regions correspond to one-tenth of a standard deviation. In addition 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)

more pronounced. Nevertheless, STOVE produces high-quality rollouts and still 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 corresponding to moving in one of the eight 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 section 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 use 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. Even when PPO is trained on ground truth states, MCTS on our model remains comparable, indicating that our dynamics model is the main cause of sample efficiency.

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, one can see 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.

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, or Watters et al. (2017); Sanchez-Gonzalez et al. (2018), who use graph neural networks to learn the dynamics of objects from images. Janner et al. (2019) show successful planning based on learned interactions, but assume access to image segmentations. Unsupervised approaches such as Jaques et al. (2019); Wu et al. (2016; 2015) address the problem by fitting the parameters of a physics engine to the observed data. For this, it is necessary to specify 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.

Only a small number of works address this by incorporating objects into unsupervised video models. Notable exceptions are SQAIR (Kosiorek et al., 2018), R-NEM (Van Steenkiste et al., 2018), and DDPAE (Hsieh et al., 2018). 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 with 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 considered in this paper.

5 Conclusion

We introduced STOVE: a structured, object-aware model for unsupervised video modeling and model-based planning. Our model 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 showed an extension of the video learning framework to the RL setting. The results demonstrate that our model of the environment 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, our model 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. Additionally, we do not currently use object appearances for tracking, which may be crucial in more complex environments such as Atari games. Finally, the availability of high-quality learned video models opens the door to the use of more sophisticated model-based RL algorithms on visual domains. In particular, by combining MCTS with a policy network, a system similar to AlphaGo (Silver et al., 2016) may be constructed, except on a learned world model instead of a known environment.

REFERENCES

Mohammad Babaeizadeh, Chelsea Finn, Dumitru Erhan, Roy H Campbell, and Sergey Levine. Stochastic variational video prediction. In *Proceedings of ICLR*, 2017.

Peter Battaglia, Razvan Pascanu, Matthew Lai, Danilo Jimenez Rezende, et al. Interaction networks for learning about objects, relations and physics. In *Proceedings of NeurIPS*, pp. 4502–4510, 2016.

Peter W Battaglia, Jessica B Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, et al. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*, 2018.

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

Christopher P Burgess, Loic Matthey, Nicholas Watters, Rishabh Kabra, Irina Higgins, Matt Botvinick, and Alexander Lerchner. Monet: Unsupervised scene decomposition and representation. *arXiv preprint arXiv:1901.11390*, 2019.

Junyoung Chung, Kyle Kastner, Laurent Dinh, Kratarth Goel, Aaron C Courville, and Yoshua Bengio. A recurrent latent variable model for sequential data. In *Proceedings of NeurIPS*, pp. 2980–2988, 2015.

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

Martin Engelcke, Adam R Kosiorek, Oiwi Parker Jones, and Ingmar Posner. Genesis: Generative scene inference and sampling with object-centric latent representations. *arXiv preprint arXiv:1907.13052*, 2019.

SM Ali Eslami, Nicolas Heess, Theophane Weber, Yuval Tassa, David Szepesvari, Geoffrey E Hinton, et al. Attend, infer, repeat: Fast scene understanding with generative models. In *Proceedings of NeurIPS*, pp. 3225–3233, 2016.

Katerina Fragkiadaki, Pulkit Agrawal, Sergey Levine, and Jitendra Malik. Learning visual predictive models of physics for playing billiards. *arXiv preprint arXiv:1511.07404*, 2015.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Proceedings of NeurIPS*, pp. 2672–2680, 2014.

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

Jun-Ting Hsieh, Bingbin Liu, De-An Huang, Li F Fei-Fei, and Juan Carlos Niebles. Learning to decompose and disentangle representations for video prediction. In *Proceedings of NeurIPS*, pp. 517–526, 2018.

Michael Janner, Sergey Levine, William T. Freeman, Joshua B. Tenenbaum, Chelsea Finn, and Jiajun Wu. Reasoning about physical interactions with object-centric models. In *Proceedings of ICLR*, 2019.

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

Lukasz Kaiser, Mohammad Babaeizadeh, Piotr Milos, Blazej Osinski, Roy H. Campbell, Konrad Czechowski, Dumitru Erhan, Chelsea Finn, Piotr Kozakowski, Sergey Levine, Ryan Sepassi, George Tucker, and Henryk Michalewski. Model-based reinforcement learning for atari. *CoRR*, 2019.

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

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

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

Adam Kosiorek, Hyunjik Kim, Yee Whye Teh, and Ingmar Posner. Sequential attend, infer, repeat: Generative modelling of moving objects. In *Proceedings of NeurIPS*, pp. 8606–8616, 2018.

Rahul G. Krishnan, Uri Shalit, and David Sontag. Deep kalman filters. *arXiv preprint arXiv:1812.08434*, 2015.

Junhyuk Oh, Xiaoxiao Guo, Honglak Lee, Richard L Lewis, and Satinder Singh. Action-conditional video prediction using deep networks in atari games. In *Proceedings of NeurIPS*, pp. 2845–2853, 2015.

Alvaro Sanchez-Gonzalez, Nicolas Heess, Jost Tobias Springenberg, Josh Merel, Martin Riedmiller, Raia Hadsell, and Peter Battaglia. Graph networks as learnable physics engines for inference and control. In *Proceedings of ICML*, 2018.

Adam Santoro, David Raposo, David G Barrett, Mateusz Malinowski, Razvan Pascanu, Peter Battaglia, and Timothy Lillicrap. A simple neural network module for relational reasoning. In *Proceedings of NeurIPS*, pp. 4967–4976, 2017.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484, 2016.

Karl Stelzner, Robert Peharz, and Kristian Kersting. Faster attend-infer-repeat with tractable probabilistic models. In *Proceedings of ICML*, pp. 5966–5975, 2019.

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

Sjoerd Van Steenkiste, Michael Chang, Klaus Greff, and Jürgen Schmidhuber. Relational neural expectation maximization: Unsupervised discovery of objects and their interactions. In *Proceedings of ICLR*, 2018.

Antonio Vergari, Robert Peharz, Nicola Di Mauro, Alejandro Molina, Kristian Kersting, and Floriana Esposito. Sum-product autoencoding: Encoding and decoding representations using sum-product networks. In *Proceedings of AAAI*, 2018.

Nicholas Watters, Daniel Zoran, Theophane Weber, Peter Battaglia, Razvan Pascanu, and Andrea Tacchetti. Visual interaction networks: Learning a physics simulator from video. In *Proceedings of NeurIPS*, pp. 4539–4547, 2017.

Nicholas Watters, Loic Matthey, Matko Bosnjak, Christopher P Burgess, and Alexander Lerchner. Cobra: Data-efficient model-based rl through unsupervised object discovery and curiosity-driven exploration. *arXiv preprint arXiv:1905.09275*, 2019.

Jiajun Wu, Ilker Yildirim, Joseph J Lim, Bill Freeman, and Josh Tenenbaum. Galileo: Perceiving physical object properties by integrating a physics engine with deep learning. In *Proceedings of NeurIPS*, pp. 127–135, 2015.

Jiajun Wu, Joseph J Lim, Hongyi Zhang, Joshua B Tenenbaum, and William T Freeman. Physics 101: Learning physical object properties from unlabeled videos. In *Proceedings of BMVC*, 2016.

Cheng Zhang, Judith Bütepage, Hedvig Kjellström, and Stephan Mandt. Advances in variational inference. *CoRR*, 2017.

Jie Zhou, Ganqu Cui, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, and Maosong Sun. Graph neural networks: A review of methods and applications. *arXiv preprint arXiv:1812.08434*, 2018.

A Appendix

A.1 Reconstructions: Sprites Data

SuPAIR does not need a latent description of the objects' appearances. Object reconstructions can be obtained by using linear-time, approximate MPE (most probable explanation) in the sum-product networks as described in Vergari et al. (2018). We follow the AIR approach and reconstruct each object separately and paste it to 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 SuPAIR 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 more complex object shapes.

A.2 Model Details

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

In the recognition network, the inferred distributions from the image model $q(z_t \mid z_{t-1})$ and physics prediction $q(z_t \mid z_{t-1})$ are multiplied to yield a single latent state distributions. Since both q-distributions are Gaussian, the product is again Gaussian, where mean and standard deviation are given by

$$\begin{aligned}
z_t &\sim q(z_t \mid z_{t-1}) \cdot q(z_t \mid x_t) \\
q(z_t \mid x_t, z_{t-1}) &\propto \mathcal{N}(\cdot \mid \mu_{t,p};\, \sigma_{t,p}) \cdot \mathcal{N}(\cdot \mid \mu_{t,d};\, \sigma_{t,d}) \\
&= \mathcal{N}(\cdot \mid \mu_t;\, \sigma_t) \\
\mu_t &= \frac{\sigma_{t,d}^2\, \mu_{t,p} + \sigma_{t,p}^2\, \mu_{t,d}}{\sigma_{t,d}^2 + \sigma_{t,p}^2} \\
\frac{1}{\sigma_t^2} &= \frac{1}{\sigma_{t,d}^2} + \frac{1}{\sigma_{t,p}^2}\,.
\end{aligned}$$

A.3 Details on Reward Prediction and Action Conditioning

To predict the influence of an action and the resulting reward, the action was added as an input. The reward is predicted by a small MLP, which is added after obtaining a high-dimensional encoding of the interactions, and optimized via MSE. During training, a random and a MCTS based policy where used to obtain samples for the model. Since the task does not contain any unseen information and the interactions of the objects are invariant to their position and other details, no specific policies for a full state exploration where needed.

A.4 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 32x32 for the billiards data and 50x50 for the gravity data. All models for video prediction were learned on grayscale data. The balls were initialised with random positions and velocities, and 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 modelled 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 a 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.

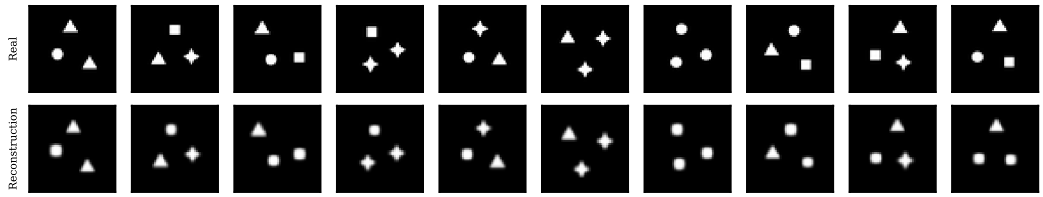

Figure 5: Reconstructions obtained from our image model when using more complex shapes.

A.5 Baselines for the physics model

Following Kosiorek et al. (2018), we looked at different hyperparameter configurations for VRNNs. We investigated the test set performance of VRNN with varying size of the hidden and latent states $[h, z]$. We tried out $[h, z]$ values of $[256, 16]$, $[512, 32]$, $[1024, 64]$, and $[2048, 32]$. For us, increases in model capacity beyond $[512, 32]$ did not yield large increases in model performance, which is why we chose $[512, 32]$ as our VRNN standard configurations. 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, we fixed the numer of objects for SQAIR. 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 obtain reconstructions of the full model state from our model. 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 frames, they stick to it.

Since our core takes as input only object positions and velocities, and not some abstract, unstructured latent state, 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.

A.6 Baselines for the reinforcement learning model

The MCTS implementation is a simple approach using the basic UCT formulation for exploration/exploitation. The c parameter is set to $1.$ in all our experiments. Since the model does not provide a natural endpoint, we cut off all searches at a depth of 20 timesteps, which means no model can predict a situation further in the future. We found this to be a good trade-off between runtime and accuracy.

When evaluating the model with MCTS, we expand each node by predicting all actions simultaneously and compute a rollout for each resulting position. This enables a better utilization of the model, since it reduces the amount of CPU-GPU data transfers. To estimate the node value function, the average of all rollouts is propagated back to the root, and each node's visitation counter is increased by 1. Furthermore, we discount the reward predicted by the model with a exponential factor of 0.95 to account for the higher uncertainty of longer rollouts. This is not done in the baseline running on the real environment, since the task contains no stochastic elements and therefore the whole rollout is deterministic and certain.

For PPO, we employ a simple 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 having 32 output filters with a kernel size of 4 and 3 respectively and a stride of 2. The MLP consists of two fully conected 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, the linking layer employs 512 and for the MLP 64 hidden units. All previously mentioned layers use rectified linear activations. The actor head then 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 different 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 used value loss weight corresponds to 5×10^{-1} and the entropy coefficient is set to 1×10^{-2}.