Capturing Future State Distributions for Effective Planning in Stochastic Environments

David Millard **Karl Pertsch** **Aleksei Petrenko** **Shao-Hua Sun**
University of Southern California
{dmillard, pertsch, petrenko, shaohuas}@usc.edu

Abstract

Intelligent agents are able to anticipate how the distribution of states in a stochastic environment evolves over time as well as create and adjust plans accordingly. Yet, existing learning algorithms either cannot leverage predictive models in stochastic environments or require extensive sampling to capture the distribution of possible future states. We propose a novel stochastic future prediction framework that predicts the evolution of the state distributions over time without requiring sampling. The learned predictive representation can be used to augment Reinforcement Learning (RL) agents. We demonstrate that the predictions of our method are able to capture uncertainty over time in a simulated stochastic environment and provide higher accuracy than those of a deterministic predictive model. We also demonstrate the effectiveness of model-based Reinforcement Learning (RL) approaches on a challenging task in the same stochastic environment and thereby show the potential of future work on combining our predictive model and RL agents.

1 Introduction

Humans are capable of anticipating the future in a variety of scenarios. This ability allows us to not only accelerate the process of learning but also create and adjust plans. Instead of producing an unimodal prediction, we are able to construct multimodal future outcomes to handle the uncertainty of environments. For instance, imagine navigating through heavy traffic as an example of a stochastic environment: being able to predict what other drivers could do based on their directional signals or acceleration allows us to smoothly change lanes and adjust our speed accordingly. Our ability to predict a whole distribution of possible outcomes enables us to anticipate potentially hazardous situations while choosing actions that lead to desirable and safe driving behaviour.

Similarly, can machines anticipate multimodal future in order to effectively learn and make plans in a stochastic environment? The learning of predictive models has attracted increasing attention in recent years [21, 25, 27, 6, 17, 4, 26]. Recent approaches focus on capturing the stochasticity in possible future predictions [29, 1, 9, 19]. Yet, those models require repeated sampling at every time step to obtain a distribution over future states. Another line of work concentrates on incorporating such predictive models into Reinforcement Learning (RL) agents to improve their decision making through prediction of action effects [22, 15, 28, 2]. However, these models either cannot handle stochastic environments [22] or require heuristics on extensive sampling schemes to capture the uncertainty in future outcomes [15, 28, 2]. To overcome these issues, we aim to develop a framework that is able to predict how *the distribution of states* evolves over time without sampling. The goal is to enable model-based RL agents to capture uncertainty in stochastic environments in a principled manner.

To this end, we propose a novel future prediction framework which implicitly captures the temporal evolution of the distribution of environment states given an initial state. By augmenting the model-free agent with a single rollout of these distributions instead of sampled trajectories we can potentially

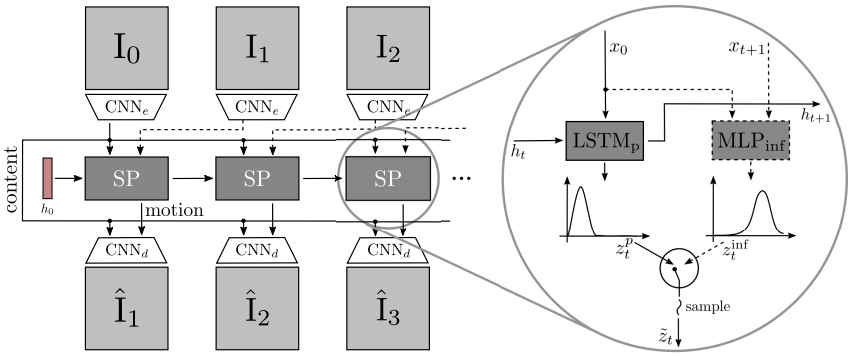


Figure 1: Overview of our state distribution prediction model (here: with image input). Review Section 3 for details of the architecture. Dashed lines indicate data flow and components only used during training. The switch in the Stochastic Prediction (SP) module samples the latent variable $\tilde{z}_t$ from the inference distribution z_t^{inf} during training and from the prior distribution z_t^p at test time.

enable RL agents to effectively utilize predictive capabilities in a stochastic environment without requiring sampling.

We conduct experiments in a simulated stochastic environment and prove that the predictions of our model accurately capture the temporal evolution of the state distribution (Section 4.1). Additionally, we show the effectiveness of model-based reinforcement learning agents on a challenging sequential decision-making task in this stochastic environment (Section 4.2), highlighting the potential of future work on the combination of our predictive model and model-based RL agents.

2 Related Work

Variational Future Prediction. A common approach for learning predictive models is through self-supervised prediction of future observations in time series data. A great body of work concentrated on the prediction of future video frames as data is abundant and the pixel-wise reconstruction loss provides dense supervisory signal [21, 25, 27, 6, 17, 4, 26]. While these models feature a deterministic prediction model the true nature of the future is multimodal, i.e. multiple possible future predictions are valid given a single sequence of conditioning frames. More recent work therefore predicts distribution over possible future frames by sampling from a latent variable [29, 1, 9, 19]. While these methods capture step-wise uncertainty but require sampling from the state distribution at every time step our model captures the evolution of the state distribution over multiple time steps into the future without requiring any sampling.

Predictive Models for Planning and Control. There are multiple approaches for incorporating predictive models in planning and control pipelines. Predicted states can be used directly to infer future reward [10, 13], to drive visual servoing [12, 11], or to enable planning through backpropagation of gradients [3, 24, 23]. Alternatively, predicted state trajectories can be integrated into an end-to-end differentiable policy pipeline [22, 2]. Another line of work extends predictive models for control to explicitly model the uncertainty in the prediction in stochastic or partially observable environments, either with Gaussian distributions over future states [7, 8, 14] or ensembles of predictive methods [5]. Recent work incorporates step-wise stochastic predictive models with deep encoder and decoder architectures into end-to-end trainable RL agents [15, 28, 2]. In contrast, we contribute a predictive model which learns a representation of future state distribution trajectories, rather than step-by-step rollouts or single-step distribution predictions. This representation can enable RL agents to capture the distribution of possible future states over time without sampling.

3 Method

We develop a model that implicitly captures the *distribution* over possible future states of the environment over time given a conditioning input state and is able to sample individual states from this distribution. An overview of our architecture is given in Figure 1.

Our model captures the uncertainty of future states within a learned prior distribution $p_\eta(z_t|x_0, t)$ parameterized by η that is conditioned on the initial system state x_0 and the current timestep t. It is trained by maximizing the log-likelihood of the predictive generative model $x_{t+1} \sim p_\theta(x_{t+1}|x_0, \tilde{z}_t)$ with parameters θ that decodes a sample $\tilde{z}_t \sim p_\eta(z_t|x_0, t)$ from the prior distribution along with the initial state x_0 into the predicted state x_{t+1}. Note that the proposed architecture is agnostic to the modality of the input observation o_t. In our experiments we demonstrate prediction in the image domain as well as for object coordinates.

We use a learned encoder network to embed the content of an input observation in a latent input space: $x_t = E(o_t)$. The generative model is implemented using a decoder network that outputs the future observation $\hat{o}_{t+1} = D(x_0, \tilde{z}_t)$. To make the approximation of the log-likelihood of this generative model feasible we employ another network, called the *inference network* $q_\phi(z_t|x_0, x_{t+1})$ with parameters ϕ in form of a Multi-Layer Perceptron (MLP) that estimates a Gaussian approximation of the posterior of the latent variable $z_t^{\text{inf}} = \text{MLP}_{\text{inf}}(x_0, x_{t+1}) = \mathcal{N}\big(\mu(x_0, x_{t+1}), \sigma(x_0, x_{t+1})\big)$. Because our model does not have access to the true next frame encoding x_{t+1} at test time, we train a prior network in form of a Long Short Term Memory Network (LSTM) [16]: $z_t^p = \text{LSTM}_p(x_0, h_t) = \mathcal{N}\big(\mu(x_0, h_t), \sigma(x_0, h_t)\big)$ that approximates the distribution over latent variables z_t by minimizing the KL-divergence between z_t^p and z_t^{inf}. The LSTM also updates its internal state $h_{t+1} = \text{LSTM}_p(x_0, h_t)$. The initial state h_0 is the output of a recurrent encoding network LSTM_e that captures the content of the K conditioning observations $o_{-(K-1):0}$. We do not need to explicitly condition the prior network on the time step as the recurrent structure of the LSTM captures the temporal progression.

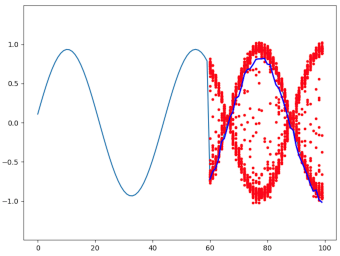


Figure 2: Bi-modal sinewave prediction. **Blue**: ground truth. **Red**: sampled predictions,

Similar to [9] our model is optimized by maximizing the log-likelihood of the generative model while regularizing the learned representation with a KL convergence constraint between the learned prior and the inference distribution:

$$\text{L}_{\theta,\phi,\eta}(x_0, x_{t+1}, t) = \mathbb{E}_{q_\phi(z_t|x_0,x_{t+1})} \log p_\theta(x_{t+1}|x_0, \tilde{z}_t) - \beta D_{\text{KL}}\big(q_\phi(z_t|x_0, x_{t+1})||p_\eta(z_t|x_0, t)\big) \quad (1)$$

The first term of the loss function, the reconstruction loss, encourages the network to embed information about the future state into the learned representation z through the inference network. In contrast, the regularization term forces the inference distribution to be predictable by the prior network that does not have access to the future state, i.e. it minimizes the information that z contains about the future state. Together the terms encourage our model to learn a representation that contains the predictive information but is minimal in terms of the mutual information $I\big(z_t, (x_0, x_{t+1})\big)$. The trade-off between the two conflicting goals is determined by the coefficient β.

Toy experiment. To verify that our proposed model is able to capture multi-modal future state distributions, we train and evaluate it on a bi-modal sine wave prediction task (see Figure 2). The first 60 samples of a sinusoidal function (blue) are observed and the following 40 samples are flipped upside down with 50% probability. Samples from the prior distribution of our model (red) successfully capture the bi-modal distribution.

4 Experiments

Environment. We introduce the bouncing ball environment in which an simulated agent tries to reach multiple goals while avoiding collisions with balls which bounce off the walls of the frame (see Figure 4). While the balls move on a straight line in between bounces we add variable amounts of Gaussian noise to the bouncing angle resulting in an environment with adjustable stochasticity. The agent receives positive reward for reaching goals and negative reward upon collision with any ball which ends the episode. At every point during an episode a fixed number of goals is present in the scene, new goals get generated at random positions when the agent reaches a goal.

Model	DETERMINISTIC			STOCHASTIC (OURS)		
Variance	0	5	10	0	5	10
PSNR	20.162	18.141	16.048	25.704	21.292	19.213
SSIM	0.93	0.915	0.897	0.933	0.912	0.899

Table 1: Performance of predictive modules. Our stochastic model consistently outperforms its deterministic counterpart across different level of stochasticity on both PSNR and SSIM, showing the importance of modeling the stochasticity.

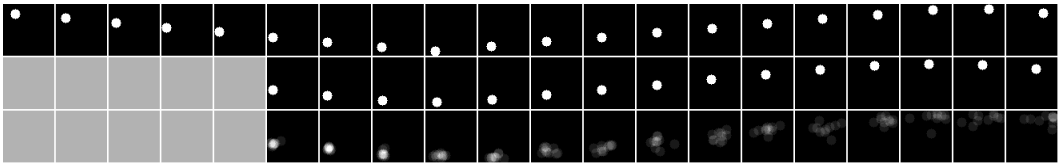


Figure 3: Prediction sequences from the bouncing ball environment. **Top**: Ground truth sequence. **Middle**: Deterministic prediction model. **Bottom**: Overlay of multiple prior samples from our model, which capture the distribution of possible future ball positions. Predicted videos: sites.google.com/view/state-dist-prediction.

4.1 Predictive Module

We train and evaluate our model and a deterministic baseline without latent variable z and KL divergence in the bouncing ball environment. Given a set of ball coordinates $\{x_1^t, y_1^t, ...x_K^t, y_K^t\}_{t=1,...,T}$, where K denotes the number of balls and T denotes the number of given time steps, our model is trained to predict $\{x_1^t, y_1^t, ...x_K^t, y_K^t\}_{t=T+1,...}$ by optimizing Equation (1). We use two standard evaluation metrics for images – Peak Signal to Noise Ratio (PSNR) and Structural Similarity Index (SSIM), to quantitatively measure the quality of the predictions. The quantitative results are shown in Table 1 and the qualitative results are shown in Figure 3. We show predicted videos of our method at sites.google.com/view/state-dist-prediction.

Our stochastic coordinate-based model consistently outperforms its deterministic counterpart across different level of stochasticity on both PSNR and SSIM. This demonstrates the importance of explicitly modeling the stochasticity. Unlike our model producing sharp results, the deterministic model generates blurry balls. We also evaluate our model with the baseline on the environment with multiple balls and different numbers of samples, which are shown in Section D.

4.2 Model-free RL Module

Since our ultimate goal is to utilize the predictive model to improve the performance of RL agents, we have conducted experiments and implemented RL baselines that pave a way towards leveraging stochastic predictive models for RL policies in non-deterministic environments. We explored different RL policy designs and evaluated our agents in the stochastic bouncing ball environment.

We train a synchronous version of the Advantage Actor-Critic (A2C) [20] on the low-dimensional input consisting of the coordinates of the agent, balls, and bonuses as input, and learn to collect as many bonuses as possible while avoiding collisions with the balls. To make the policy invariant to the order of balls and goals, we apply an idea similar to [30]: we independently encode all of the objects and then aggregate them using a symmetric operation (*e.g.* summation). The results are shown in Section C, videos of the agent acting can be found at sites.google.com/view/state-dist-prediction.

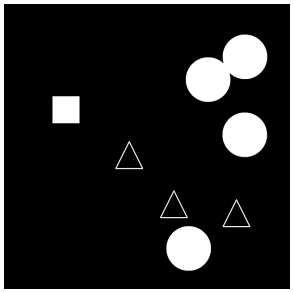


Figure 4: RL Environment. The agent (square) has to avoid the bouncing obstacles (circles) while collecting bonuses (triangles).

4.2.1 I2A Architecture with Ground Truth Prediction

To more effectively leverage the information about future states, we implement the imagination-augmented agents model (I2A) [22] which introduces model-free and model-based paths in the policy architecture. I2A shows significantly better performance compared to our A2C baseline (Figure 6), which proves the potential of our future research direction: combining our predictive model with a model-free RL policy.

5 Discussion

Our experiments have shown that the proposed model can capture future state distributions and that RL agents benefit from information about future states. Progressing towards combining our model and RL agents we draw the main conclusion that it is vital to (i) condition the prediction on the agent's action and (ii) also model future rewards along with future states. As a next step we plan to extend the predictive model to capture the joint distribution of future state, agent action and reward.

References

[1] Mohammad Babaeizadeh, Chelsea Finn, Dumitru Erhan, Roy H. Campbell, and Sergey Levine. Stochastic variational video prediction. In *ICLR*, 2018.

[2] Lars Buesing, Theophane Weber, Sebastien Racaniere, S. M. Ali Eslami, Danilo Jimenez Rezende, David P. Reichert, Fabio Viola, Frederic Besse, Karol Gregor, Demis Hassabis, and Daan Wierstra. Learning and querying fast generative models for reinforcement learning. *arXiv preprint arXiv:1802.03006*, 2018.

[3] Arunkumar Byravan, Felix Leeb, Franziska Meier, and Dieter Fox. Se3-pose-nets: Structured deep dynamics models for visuomotor control. In *ICRA*, 2018.

[4] Silvia Chiappa, Sebastien Racaniere, Daan Wierstra, and Shakir Mohamed. Recurrent environment simulators. In *ICLR*, 2017.

[5] Kurtland Chua, Roberto Calandra, Rowan McAllister, and Sergey Levine. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. *arXiv preprint arXiv:1805.12114*, 2018.

[6] Francesco Cricri, Xingyang Ni, Mikko Honkala, Emre Aksu, and Moncef Gabbouj. Video ladder networks. *arXiv preprint arXiv:1612.01756*, 2016.

[7] Marc Deisenroth and Carl E Rasmussen. Pilco: A model-based and data-efficient approach to policy search. In *ICML*, 2011.

[8] Marc Peter Deisenroth, Dieter Fox, and Carl Edward Rasmussen. Gaussian processes for data-efficient learning in robotics and control. *IEEE transactions on pattern analysis and machine intelligence*, 2015.

[9] Emily Denton and Rob Fergus. Stochastic video generation with a learned prior. In *ICML*, 2018.

[10] Alexey Dosovitskiy and Vladlen Koltun. Learning to act by predicting the future. In *ICLR*, 2017.

[11] Frederik Ebert, Chelsea Finn, Alex X. Lee, and Sergey Levine. Self-supervised visual planning with temporal skip connections. In *CoRL 2017*, 2017.

[12] Chelsea Finn and Sergey Levine. Deep visual foresight for planning robot motion. In *ICRA*, 2017.

[13] Katerina Fragkiadaki, Pulkit Agrawal, Sergey Levine, and Jitendra Malik. Learning visual predictive models of physics for playing billiards. In *ICLR*, 2016.

[14] Yarin Gal, Rowan McAllister, and Carl Edward Rasmussen. Improving pilco with bayesian neural network dynamics models. In *Data-Efficient Machine Learning workshop, ICML*, 2016.

[15] David Ha and Juergen Schmidhuber. Recurrent world models facilitate policy evolution. In *NIPS*, 2018.

[16] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. MIT Press, 1997.

[17] Nal Kalchbrenner, Aaron van den Oord, Karen Simonyan, Ivo Danihelka, Oriol Vinyals, Alex Graves, and Koray Kavukcuoglu. Video pixel networks. In *ICML*, 2017.

[18] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

[19] Alex X Lee, Richard Zhang, Frederik Ebert, Pieter Abbeel, Chelsea Finn, and Sergey Levine. Stochastic adversarial video prediction. *arXiv preprint arXiv:1804.01523*, 2018.

[20] Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Timothy P. Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *ICML*, 2016.

[21] Junhyuk Oh, Xiaoxiao Guo, Honglak Lee, Richard L Lewis, and Satinder Singh. Action-conditional video prediction using deep networks in atari games. In *NIPS*, 2015.

[22] Sebastien Racaniere, Theophane Weber, David Reichert, Lars Buesing, Arthur Guez, Danilo Jimenez Rezende, Adria Puigdomenech Badia, Oriol Vinyals, Nicolas Heess, Yujia Li, Razvan Pascanu, Peter Battaglia, Demis Hassabis, David Silver, and Daan Wierstra. Imagination-augmented agents for deep reinforcement learning. In *NIPS*. 2017.

[23] 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. *arXiv preprint arXiv:1806.01242*, 2018.

[24] Aravind Srinivas, Allan Jabri, Pieter Abbeel, Sergey Levine, and Chelsea Finn. Universal planning networks: Learning generalizable representations for visuomotor control. In *ICML*, 2018.

[25] Nitish Srivastava, Elman Mansimov, and Ruslan Salakhudinov. Unsupervised learning of video representations using lstms. In *ICML*, 2015.

[26] Sergey Tulyakov, Ming-Yu Liu, Xiaodong Yang, and Jan Kautz. Mocogan: Decomposing motion and content for video generation. In *CVPR*, 2018.

[27] Carl Vondrick, Hamed Pirsiavash, and Antonio Torralba. Generating videos with scene dynamics. In *NIPS*, 2016.

[28] Greg Wayne, Chia-Chun Hung, David Amos, Mehdi Mirza, Arun Ahuja, Agnieszka Grabska-Barwinska, Jack Rae, Piotr Mirowski, Joel Z Leibo, Adam Santoro, et al. Unsupervised predictive memory in a goal-directed agent. *arXiv preprint arXiv:1803.10760*, 2018.

[29] Tianfan Xue, Jiajun Wu, Katherine Bouman, and Bill Freeman. Visual dynamics: Probabilistic future frame synthesis via cross convolutional networks. In *NIPS*, 2016.

[30] Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabás Póczos, Ruslan Salakhutdinov, and Alexander J. Smola. Deep sets. In *NIPS*, 2017.

A Detailed Architectures

A.1 Predictive Model

For coordinate observations we use multi-layer perceptrons (MLPs) as encoder and decoder with three layers with $[8, 16, 32]$ neurons respectively. In case of image observations we instead use convolutional neural networks (CNNs) for encoder and decoder with skip connections between the features of the last conditioning image encoding and each future output decoding. The both networks use a stride of 2 (down- or upsampling respectively), kernel-size 3x3, leaky ReLU activations and $[8, 16, 32]$ channels in the three layers respectively. For both coordinate and image observations an additional fully-connected layer without activation function maps the output of encoder to the latent observation encoding space of size 32.

Both the prior network LSTM_p and the conditioning sequence encoder LSTM_e have a state size of 32. The former outputs 16×2 values for mean and variance values of the 16-dimensional predictive latent distribution z_t^p. The latter outputs the initial state of the prior network. We implement the inference network MLP_{inf} with a three-layer perceptron with $[96, 64, 32]$ neurons in the respective layer. Finally we copy this MLP architecture (but with independent weights) to map the sampled z-values $\tilde{z}_t$ to the output latent encoding that is passed to the decoder.

A.2 RL agents

The pixel-based RL agent is based on a 5-layer CNN with kernel size of 5 in the first layer, followed by layers with kernel size 3. We used layers with $[32, 64, 128, 128, 128]$ channels respectively, the stride of 2 for all layers, ReLU activations and no skip connections. The CNN is followed by a fully-connected layer with 256 neurons, after which the network splits into two heads, one for value estimation and one for action logits, each of them contains another FC layer with 128 neurons. To capture the dynamics of the environment we stack three environment frames $(t, t-1, t-2)$ as input.

The baseline coordinate-based policy is the same as the pixel-based policy, except the CNN encoder is replaced with a fully-connected encoder for every input frame. The FC encoder weights are shared between the current and the past frames. The DeepSets-inspired architecture leverages a separate 2-layer FC encoders for balls, bonuses and the agent. The features of objects of the same type are aggregated with summation and fed into the same MLP network $[256, 128 + 128]$.

Finally, the I2A agent uses the same architecture as the "DeepSets" network for it's model-free path and for the distilled policy. Model-based path is represented by a single 128-cell LSTM layer that is responsible for the encoding of the future rollouts. The output of the last LSTM step is concatenated with the second-to last FC layer in the model-free network.

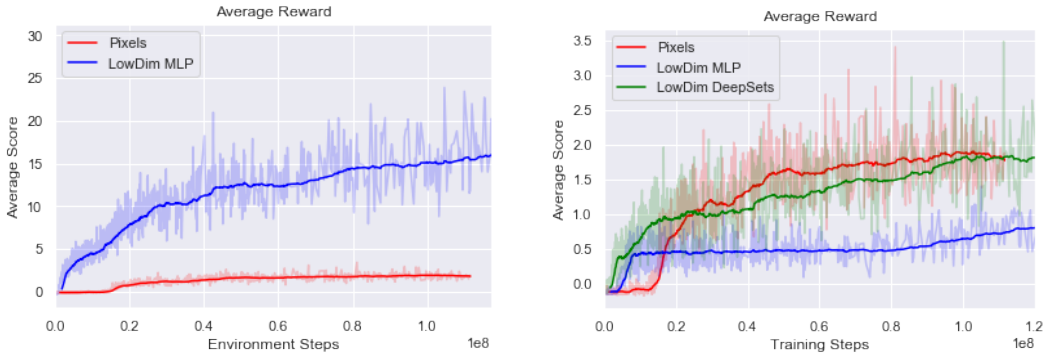
B Training Details

B.1 Predictive Model

We train our network using the Adam optimizer [18], with $\beta_1 = 0.99$ and $\beta_2 = 0.999$, and a learning rate of 1×10^{-4} for the generator. We use a batch size of 32, condition on 5 input observations and predict 25 time steps into the future. We optimize mean-squared-error loss on both coordinate values and image pixel values. We set the trade-off parameter from Equation (1) to $\beta = 1 \times 10^{-3)}$.

B.2 RL agents

All of the agents were trained with A2C algorithm with the following parameters: 32 workers and 5-step TD rollout for a total of 160 observations in a batch, learning rate of 1×10^{-4}, entropy penalty coefficient is 0.002. We use Adam optimizer with default parameters. In the I2A arcitecture, the distilled policy is trained with the same optimizer using the softmax crossentropy loss between the distilled policy action logits and the I2A policy action probabilities. We use the stop gradient operator on the main policy action probabilities to make sure that training of the distilled policy does not influence the main policy.



(a) Performance on the Bouncing Balls environment with a single ball and a single goal. Low-dimensional policy significantly outperforms pixel observations.

(b) Performance on the (much harder) version of Bouncing Balls environment with four balls and three goals. A DeepSets approach to order-invariance significantly improves performance.

Figure 5: In coordinate-based version of the Bouncing Balls environment the order-invariance provided by DeepSets approach allows the MLP policies to match the performance of pixel-based CNN policies.

C Reinforcement Learning Baselines

As a model-free baseline, we first train the RL agent using the synchronous version of the Advantage Actor-Critic (A2C) algorithm [20]. It is able to achieve good performance in a pixel-based version of the environment, the coordinate-based (low-dimensional) version described in Section 4.1 has proven to be a substantial challenge. The learning curves in Figure 5 provide an intuition why this might be the case. When the environment contains only one instance of every object (agent, obstacle, goal), the coordinate-based version provides pretty much the perfect object features. In this case the low-dimensional policy parameterized by an MLP learns much faster (Figure 5a) than the pixel-based agent, mainly because the CNN needs many environment samples to learn the visual features that would represent the relative positions of objects.

However, when we applied the MLP architecture to the final version of the environment (Figure 4) we were not able to match the performance of the CNN A2C policy (Figure 5b). The reason for that is the inherent advantages of the CNN architecture, namely invariant to the order of the objects. Different obstacles and goals are visually indistinguishable, therefore CNN is able to learn a single set of filters to handle a broad range of situations in the environment. On the other hand, the MLP architecture needs to learn a ruleset for every combination and permutation of the objects, hence solving the problem that is combinatorially more complex.

Figure 6: For the same wall clock time, the I2A agent augmented with ground-truth model outperforms the model-free baseline. I2A with oracle model is order-of-magnitude more sample efficient.

Model	Deterministic			Stochastic (ours)		
Variance	0	5	10	0	5	10
PSNR	20.284	18.367	17.485	18.293	16.524	15.987
SSIM	0.891	0.787	0.725	0.929	0.908	0.906

Table 2: Performance with image-based predictions.

Model	Stochastic (ours)			
# of balls	1	2	3	4
PSNR	21.292	12.784	10.632	8.724
SSIM	0.912	0.823	0.747	0.65

Table 3: Performance on sequences with multiple balls.

To counteract this we applied the architecture known as DeepSets [30]. In the DeepSets network the features of all of the objects are generated by the same encoder with the shared weights and then aggregated using a symmetric operation, *e.g.* we use summation. The DeepSets architecture allows us to learn a robust coordinate-based policy that matches the performance of the pixel-based version, while training much faster in terms of wall clock time.

D Additional Prediction Results

Image-based prediction. We apply our prediction model to image observations. Differences in the architecture are detailed in Section A.1, qualitative results are shown in Figure 7, quantitative results in Table 2. While our model shows similar performance, both qualitative and in quantitative metrics, the deterministic baseline values for PSNR increase. This is because it now models the mean in pixel space, not in coordinate space. As a result it can roughly capture the future state distribution (see Figure 7, middle row). Yet, this result does not generalize to coordinate-based prediction where the unimodal prediction is not sufficient to capture the distriution over future states. Additionally, our model still substantially outperforms the baseline in SSIM, as it can still sample highly structured future trajectories, not just a single blurry mode.

Prediction results on sequences with multiple balls. We show that our model scales to more complicated problems that involve the prediction of muiltiple balls. Qualitative results are shown in Figure 8, quantitative results in Table 3. While the performance of model decreases when the number of balls increases, as the problem gets harder, it is still able to capture the distribution of future states. Further improvements could be achieved by increasing the capacity of the latent space or the encoder/decoder model, but these will be left for future work.

Prediction results with increasing number of prior samples. In Figure 9 we show the prediction of our model when averaging an increasing number of samples from the prior (quantitative results in Table 4). As can be seen the output image gets increasingly blurry as the average approximates the future state distribution. Conventional, sampling-based agents would need to sample an increasing number of trajectories from a predictive model to obtain a good estimate of the future state distribution as shown in Figure 9. In contrast our model captures the whole information in its prior distribution (before sampling). This shows the potential for using it to augment RL agents.

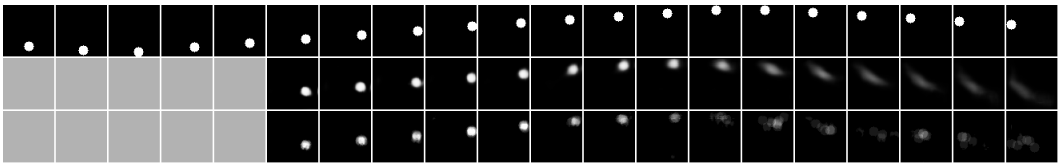


Figure 7: Image-based prediction results. **Top**: Groundtruth sequence. **Middle**: Deterministic prediction model with image input. **Bottom**: Overlay of multiple prior samples from our model with image input. In pixel space the deterministic model captures the mean in pixel values $[0 \dots 255]$ and therefore also captures information about the positional distribution. Yet, this does not hold for the coordinate-based prediction.

Model	DETERMINISTIC		
# of samples	5	50	500
PSNR	16.645	19.044	21.087
SSIM	0.895	0.899	0.902

Table 4: Performance of increasing number of prior samples.

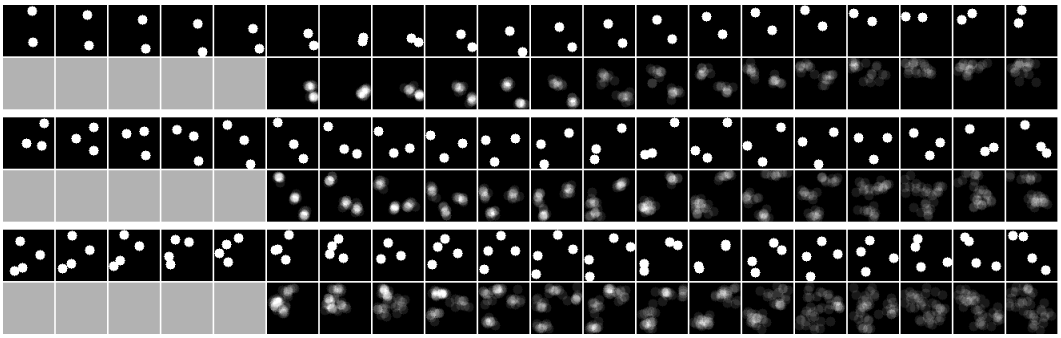


Figure 8: Prediction results on sequences with multiple balls. **For each double, Top**: Groundtruth sequence. **Bottom**: Overlay of multiple prior samples from our model.

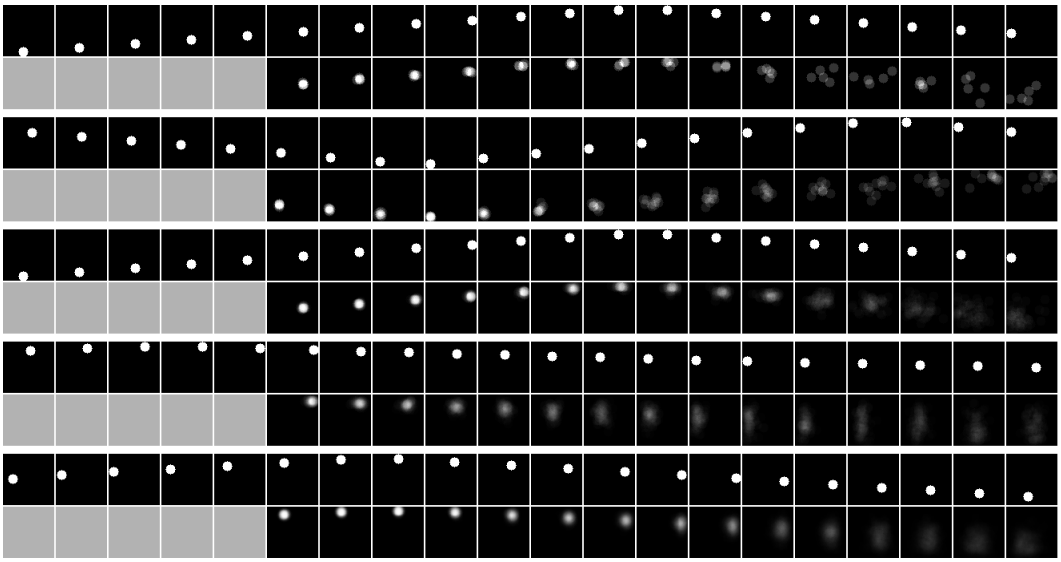


Figure 9: Prediction results with increasing number of prior samples. **For each double, Top**: Groundtruth sequence. **Bottom**: Overlay of multiple prior samples from our model. **Doubles, top to bottom**: $[5, 10, 50, 100, 500]$ prior samples. The more samples are drawn the more the method approximates the true distribution of possible future states.