Task-Agnostic Dynamics Priors for Deep Reinforcement Learning

Yilun Du [1] **Karthik Narasimhan** [2]

Abstract

While model-based deep reinforcement learning (RL) holds great promise for sample efficiency and generalization, learning an accurate dynamics model is often challenging and requires substantial interaction with the environment. A wide variety of domains have dynamics that share common foundations like the laws of classical mechanics, which are rarely exploited by existing algorithms. In fact, humans continuously acquire and use such *dynamics priors* to easily adapt to operating in new environments. In this work, we propose an approach to learn task-agnostic dynamics priors from videos and incorporate them into an RL agent. Our method involves pre-training a frame predictor on task-agnostic physics videos to initialize dynamics models (and fine-tune them) for unseen target environments. Our frame prediction architecture, SpatialNet, is designed specifically to capture localized physical phenomena and interactions. Our approach allows for both faster policy learning and convergence to better policies, outperforming competitive approaches on several different environments. We also demonstrate that incorporating this prior allows for more effective transfer between environments.

1 Introduction

Recent advances in deep reinforcement learning (RL) have largely relied on model-free approaches, demonstrating strong performance on a variety of domains (Silver et al., 2016; Mnih et al., 2013; Kempka et al., 2016; Zhang et al., 2018c). However, model-free techniques do not have good sample efficiency (Sutton, 1990) and are difficult to adapt to new tasks or domains (Nichol et al., 2018). A key reason for this is a single value function is used to represent both the agent's policy and its knowledge of environment dynamics, which can result in heavy overfitting to a particular task (Zhang et al., 2018b). On the other hand, model-based RL allows for decoupling the dynamics model from the policy, enabling better generalization and transfer across tasks (Zhang et al., 2018a). The challenge with model-based RL, however, lies in estimating an accurate dynamics model of the environment while simultaneously using it to learn a policy, often leading to sub-optimal policies and slower learning. One way to alleviate this problem is to initialize dynamics models with universal *task-agnostic* priors that allow for more efficient and stable model-based learning.

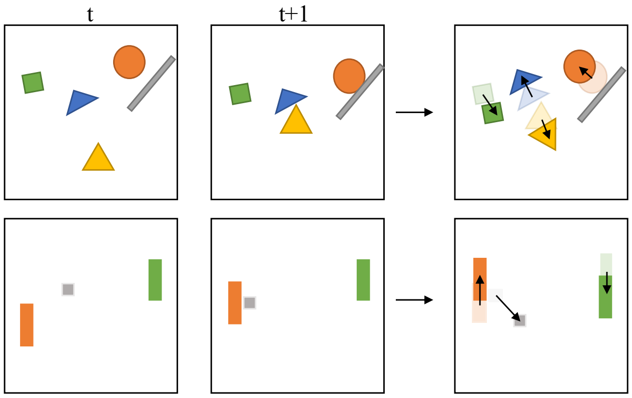

Figure 1. Two different environments with object dynamics that obey the common laws of physics (*top:* PhysWorld, *bottom:* Atari Pong). Agents that have a knowledge of general physics will be able to adapt quickly to either environment.

For example, consider learning dynamics models for the two different scenarios shown in Figure 1 (top and bottom). Both environments contain a variety of objects moving with different velocities and rotations. Current approaches require a large number of samples to learn a robust transition model of either world. For instance, in the first environment, inferring that the *orange circle* is a freely moving object will require observing the circle moving in a variety of different directions. Or to understand the laws governing elastic collisions between two bodies (e.g. the circle and the grey rectangle) requires observing several instances of collisions at various angles and velocities. On the other hand, humans have reliable priors that allow for understanding dynamics of new environments quickly (Dubey et al., 2018) – one such prior is an understanding of physical laws of motion.

[1]Massachusetts Institute of Technology (Work partially done at OpenAI) [2]Princeton University. Correspondence to: Yilun Du <yilundu@mit.com>, Karthik Narasimhan <karthikn@cs.princeton.edu>.

Proceedings of the 36^{th} International Conference on Machine Learning, Long Beach, California, PMLR 97, 2019. Copyright 2019 by the author(s).

In this work, we demonstrate that learning a task-agnostic dynamics prior (e.g. concepts like velocity, acceleration or elasticity) allows for accurate and more efficient estimation of the dynamics of new environments, resulting in better control policies.

In order to obtain a prior for physical dynamics, we perform unsupervised learning over raw videos containing moving objects. Specifically, we train a dynamics model to predict the next frame given the previous k frames, over a wide variety of scenarios with moving objects. The parameters of the model implicitly capture general laws of physics, which are useful in predicting entity movements. We initialize the dynamics model of the environment with these pre-trained parameters and fine-tune them using transitions from the specific task, while simultaneously learning a policy for the task. The dynamics model is used to predict future frames up to a finite horizon, which are then used as additional input into a policy network, similar to the approach of (Weber et al., 2017). Importantly, our frame prediction model is not action-conditional like most prior work that employs such models in reinforcement learning (Oh et al., 2015; Weber et al., 2017).

Learning a good future frame model is challenging mainly for two reasons: a) the large dimensionality of the output space with arbitrary moving objects and interactions, and b) the partial observability in environments (Mathieu et al., 2015). Prior approaches (Oh et al., 2015) suffered from error compounding since they encode the entire image into a single vector before decoding the output, thereby missing out fine-grained spatial information. Others like the ConvLSTM (Xingjian et al., 2015) are better at capturing spatio-temporal interactions but suffer from poor generalization due the use of additive update equations. To overcome these issues, we propose a new architecture (SpatialNet) that consists of a convolutional encoder, a spatial memory block, and a convolutional decoder that better captures localized dynamics. The spatial memory module operates by performing convolution operations over a temporal 3-dimensional state representation that keeps spatial information intact. This allows the network, which includes residual connections, to capture localized physics of objects such as directional movements and collisions in a fine-grained manner as well as efficiently keep track of static background information. This results in lower prediction error, better generalization and invariance to the size of inputs.

We evaluate our approach on three different RL scenarios. First, we consider PhysWorld, a suite of randomized 2D physics-focused games, where learning object movement is crucial to a successful policy. Next we consider PhysShooter3D, a 3D environment with rigid body dynamics and partial observations. Finally, we also evaluate on a stochastic variant of the popular ALE framework consisting of Atari games (Machado et al., 2017a). In all scenarios, we first demonstrate the value of learning a task-agnostic prior for model dynamics - for instance, our agent achieves up to 130% higher performance on a shooting game, PhysShooter and 56.5% higher on the Atari game of Asteroids, compared to the most competitive baseline. Further, we also show that the dynamics model fine-tuned on these tasks transfers better to new tasks. For instance, our model achieves a relative score improvement of 26.9% on transfer from PhysForage to PhysShooter (both games from PhysWorld), significantly higher than a score improvement of 5.4% using a *policy-transfer* baseline.

2 Related Work

There are two main lines of work that are closely related to this paper. The first is that of learning and using generic video prediction models for reinforcement learning (Oh et al., 2015; Finn et al., 2016; Weber et al., 2017). The key idea is to train a model to predict future frames on the target task and hallucinate additional trajectories that can help an agent learn faster. The second direction is to incorporate physics priors into parameterized dynamics models for future state prediction (Nguyen-Tuong and Peters, 2010; Kansky et al., 2017). The former path requires only pixel inputs but does not generalize well across tasks. The latter has the potential to generalize but requires manual specification of priors. Our work aims to combine the best of both worlds – learn a frame prediction model that is task-agnostic and captures an effective notion of physics to serve as a useful prior.

Video prediction models. Our frame prediction model is closest in spirit to the ConvolutionalLSTM model which has been applied to several domains (Xingjian et al., 2015; Zhu et al., 2017; Ke et al., 2017). Similar architectures that incorporate differentiable memory modules (Patraucean et al., 2015) or relational intermediates (Watters et al., 2017) have been proposed, with applications to deep RL (Parisotto and Salakhutdinov, 2017). While the ConvLSTM model is reasonably effective at predicting future frames, the additive LSTM update equations are not well suited to capture localized physical interactions.[*] Our architecture is simpler and more natural at capturing physical dynamics and entity movements – this allows for better generalization as we demonstrate in our experiments.

Several recent methods have also combined policy learning with future frame prediction in different ways. Action-conditioned frame prediction has been used to simulate trajectories for policy learning (Oh et al., 2015; Finn et al., 2016; Weber et al., 2017). Predicted frames have also been

[*] While the model theoretically can learn to ignore unnecessary operations, optimizing the parameters effectively is difficult because of a lack of proper inductive bias in the architecture.

used to incentivize exploration in agents, via hashing (Yin et al., 2017) or using the prediction error to provide intrinsic rewards (Pathak et al., 2017). The main departure of our work from these papers is that we learn a frame prediction model that is not conditioned on actions, and from videos not related to a task, which allows us to employ the model on a variety of tasks.

Parameterized physics models. Several recent papers have explored the idea of incorporating physics priors into learning dynamics models of environments (Nguyen-Tuong and Peters, 2010; Cutler et al., 2014; Cutler and How, 2015; Scholz et al., 2014; Kansky et al., 2017; Battaglia et al., 2016; Mrowca et al., 2018; Xie et al., 2016). More recent work trained an object-oriented dynamics predictor by segmenting input frames into sets of objects (Zhu et al., 2018). While all these approaches demonstrate the importance of having relevant priors to sample efficient model learning, they all require some form of manual parameterization. In contrast, we learn physics priors in the form of the parameters of a predictive neural network, only using raw videos.

Decoupling dynamics from policy. Our work also relates to previous approaches on decoupling the agent's knowledge of the environment dynamics from its task-oriented policy. Successor representations (Dayan, 1993) decompose the agent's value function into a feature-based state representation and a reward projection operator, resulting in better exploration of the state space (Kulkarni et al., 2016; Barreto et al., 2017; Machado et al., 2017b). While these state abstractions help with exploration, such representations do not explicitly capture dynamics models of the environment. More recent work has proposed approaches to learn separate models for dynamics and rewards and use it to perform online planning (Zhang et al., 2018a) or learn independently controllable factors in the environment (Thomas et al., 2017). However, these assume access to task-specific transitions, while we learn a prior from task-independent videos and demonstrate its usefulness in learning different environment dynamics.

3 Framework

Our goal is to demonstrate that acquiring task-agnostic dynamics priors from raw videos helps agents learn faster in new environments. To this end, our approach consists of two phases:

1. **Pre-training a dynamics predictor**: We first train a suitable neural network architecture to predict pixels in the next frame given the previous k frames of a video. In this work, we use videos of objects moving according to classical mechanics, without any extra annotations.
2. **Reinforcement learning**: We use the pre-trained frame predictor from the previous phase to initialize the dynamics model for an RL agent. This dynamics model is used to predict a few frames into the future, which is used as additional context for the control policy. The dynamics model is also simultaneously fine-tuned using trajectories from the environment.

We first describe how we use the frame prediction model for reinforcement learning, and then discuss different options for a frame predictor, including our new architecture, SpatialNet.

3.1 Reinforcement Learning with Dynamics Predictors

There are several ways one can incorporate a dynamics model into a reinforcement learning setup. One approach is to use the model to generate synthetic trajectories and use them in addition to observed transitions while training a policy (Oh et al., 2015; Feinberg et al., 2018). Another option is to perform rollouts from the current step using the model and then use the predicted states as additional context input to the policy (Weber et al., 2017). Our method is similar to the latter – we use our learned dynamics model to predict k future frames and concatenate these frames along with the current frame to form the input to our policy network. There are two differences however – (1) we predict future state observations *without conditioning on the actions of the agent* and without rewards since our dynamics model is task agnostic, and (2) we do not use a global encoding for future frames, but instead stack the frames and use convolution operations to extract local dynamic information.

Formally, consider a standard Markov Decision Process (MDP) setup represented by the tuple $\langle S, A, T, R\rangle$, where S is the set of all possible state configurations, A is the set of actions available to the agent, T is the transition distribution, and R is the reward function. Assuming our dynamics model to be Ω, and given the current state s_t, we first apply our prediction model iteratively to obtain future state predictions:

$$\hat{s}_{t+1} = \Omega(s_t), \hat{s}_{t+2} = \Omega(\hat{s}_{t+1}), \ \dots \ \hat{s}_{t+k} = \Omega(\hat{s}_{t+k-1})$$

We then train a policy network to output actions using all these predicted states as input in addition to the current state:

$$a_t = \pi(s_t, \hat{s}_{t+1}, \ \dots \ \hat{s}_{t+k}) \quad (1)$$

For the policy network, we follow the architecture described in (Mnih et al., 2015) and use the Proximal Policy Optimization (PPO) (Schulman et al., 2017) algorithm for learning from rewards obtained in the task. We call this agent an Intuitive Physics Agent (IPA) since it first learns an intuitive prior of physical interactions.

We update policy parameters by using the PPO loss:

$$L(\theta) = \mathbb{E}[\min(r_t(\theta)A_t, \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)A_t]$$

where $r_t = \frac{\pi_\theta(a_t|s_t,\hat{s}_{t+1},\,\dots\,\hat{s}_{t+k})}{\pi_{\theta_{\text{old}}}(a_t|s_t,\hat{s}_{t+1},\,\dots\,\hat{s}_{t+k})}$ and the advantage, A_t, is computed using the value function $V(s_t, \hat{s}_{t+1},\,\dots\,\hat{s}_{t+k})$. Simultaneously, we also update the parameters of the dynamics model using the transitions from the environment with a pixel prediction loss (described in Section 3.2.) However, policy gradients are not back-propagated to the dynamics predictor.

3.2 Dynamics Prediction

Prior work has investigated a variety of frame prediction models. LSTM-based recurrent networks (Oh et al., 2015) are not ideal for this task since they encode the entire scene into a single latent vector, thereby losing the localized spatio-temporal correlations that are important for making accurate physical predictions. On the other hand, the ConvLSTM (Xingjian et al., 2015) architecture has localized spatio-temporal correlations, but is not able to accurately maintain global dynamics of entities due to LSTM state updates and limited separation of stationary and non-stationary objects. (as also seen in our experiments in Section 4.1).

Predicting the physical behavior of an entity requires a model that can perform two crucial operations – 1) isolation of the dynamics of each entity, and 2) accurate modeling of localized spaces and interactions around the entity. In order to satisfy both desiderata, we propose a new architecture, SpatialNet, which uses a spatial memory that explicitly encodes dynamics that are updated with object movement through convolutions. This allows us to implicitly capture and maintain localized physics, such as entity velocities and collisions between entities, in our frame prediction model and results in significantly lower long term prediction error.

SpatialNet Architecture SpatialNet is conceptually simple and consists of three modules (Figure 2). The first module is a standard convolutional encoder $\mathcal{E}$ that converts an input image x_t into a 3D representational map z_t. The second module is a spatial memory block, σ, that converts z_t and the hidden state h_t from the previous timestep into an output representation o_t and new hidden state h_{t+1}. Finally, we have a convolutional decoder $\mathcal{D}$ that predicts the next frame x_{t+1} from o_t. Both the encoder and decoder modules ($\mathcal{E}$ and $\mathcal{D}$) use two convolutional layers each with residual connections.

We implement the spatial memory block σ as a 2D convolution operation. The module takes in a previous hidden state h_t and input z_t at timestep t, both of shape $k \times n \times n$ where k is the number of channels and $n \times n$ is the dimensionality of the grid. We then perform the following operations:

$$\begin{aligned} i_t = f(C_e \oplus [h_t; z_t]); \quad u_t = f(C_u \oplus [i_t; h_t]) \\ h_{t+1} = f(C_{dyn} \oplus u_t); \quad o_t = f(C_d \oplus [z_t; h_{t+1}]) \end{aligned} \tag{2}$$

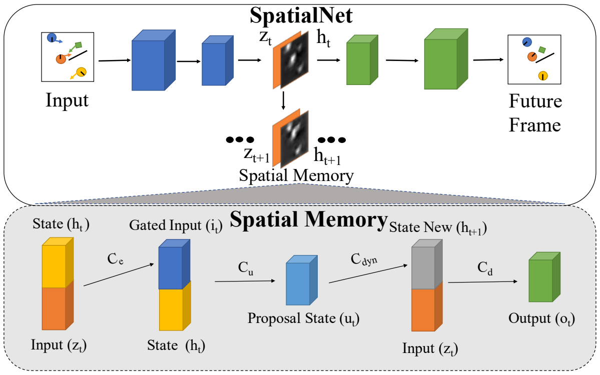

Figure 2. Overview of the SpatialNet architecture. SpatialNet takes an RBG image as input and passes it into encoder ($\mathcal{E}$) consisting of two residual blocks to form an input encoding z_t. z_t is processed by a *spatial memory* module (σ) to obtain an output representation o_t, which is used by the decoder ($\mathcal{D}$) to predict the next frame. The spatial memory stores meta information about each entity and its locality. See Section 3 for more details.

where $\oplus$ denotes a convolution, $[;]$ denotes concatenation, C_e, C_u, C_{dyn}, C_d are convolutional kernels and f is a non-linearity (we use ELU (Clevert et al., 2015)). The module first encodes a combination of z_t and h_t into a proposal state u_t, using two convolutions C_e, C_u. C_{dyn} acts like a dynamics simulator and generates a new hidden state h_{t+1}, which captures the localized predictions for the next state around each entity. Finally, C_d uses h_{t+1} and z_t to produce o_t, encoding information about the entire frame to be rendered by subsequent decoding.

Intuitively, the SpatialNet architecture biases the module towards storing relevant physics information about each entity in a block of pixels at the entity's corresponding location. This information is sequentially updated through the convolutions, while static information such as background texture is passed directly through the input encoding z_t (see Figure 5 of appendix). We note that our spatial memory is not action-conditional, which allows us to learn from task-independent videos, as well as generalize better to new environments. Given training videos $D = \left\{(x_1^{(i)}, x_2^{(i)}, \dots, x_{T_i}^{(i)}\right\}_{i=1}^{N}$, we learn the parameters of the model using a standard MSE-based loss function, $L(\theta) = \sum_i \sum_j \|\hat{x}_j^i - x_j^i\|^2$.

SpatialNet is inspired by the ConvLSTM model but is different from ConvLSTM in that while ConvLSTM performs an additive state updation operation ($c_t = f_t \cdot c_{t-1} + i_t \cdot \tanh(W_{cx}x_t + W_{hc}h_{t-1} + b_c)$), SpatialNet uses convolutions to update the hidden state (Eqn. 2). This allows SpatialNet to better simulate moving objects and physical interactions. Another difference is that SpatialNet has residual connections, which provides a more straightforward inductive bias towards maintaining both static and dynamic information across states.

Ego-dynamics One important feature of our dynamics predictor is that it is not conditioned on the action(s) of the agent, i.e. it does not account for ego-dynamics. We make this choice in order to make the dynamics prediction model task-agnostic. As we demonstrate in our experiments (Section 4.3, this makes our approach generalize well to a variety of different tasks, and learn faster in transfer experiments.

4 Experiments

We perform two empirical studies to evaluate our hypothesis. First, we evaluate various frame prediction models, including our proposed SpatialNet, in terms of their capacity to predict future states and model physical interactions (Sections 4.1 and 4.2). Then, we investigate the use of these dynamics predictors for policy learning in different environments (Section 4.3).

Physics video dataset In order to train a prediction model specifically for physical interaction, we generate a new video dataset, *PhysVideos*, using a 2-D physics engine (Pymunk). Each video in the dataset has frames of size 84 × 84 × 3 with 4-8 different shapes (such as squares or circles) moving inside a room with up to 3 randomly generated interior walls (see Figure 1 (top)). Objects are initialized with random positions and velocities, a friction coefficient of 0.9 and elasticity of 0.95, resulting in diverse object movements across each trajectory. Being able to predict the future in this type of environment requires 2-dimensional physics reasoning, such as inferring velocity from past movement, anticipating changes in momentum due to collisions, and predicting rotations of each object. We generate 5000 different trajectories in total – 4500 for training a dynamics predictor and 500 for testing – with each trajectory having a length of 125 steps. See supplementary material for sample trajectories.

4.1 Frame Prediction

In this section, we evaluate various frame prediction models on their accuracy across different horizons. We report results on the 500 trajectories from the test set of *PhysVideos*.

Baselines We compare our model, SpatialNet, with the following baselines:

1. *RCNet*: the model of (Oh et al., 2015) modified to work without action-conditioning, i.e. $h_t^{dec} = h_t^{enc}$.
2. *ConvLSTM* (Xingjian et al., 2015): this model replaces all the inner operations of an LSTM with convolutions. We use a kernel size of 5 and the same encoders and decoders as in SpatialNet.
3. *ConvLSTM + Residual*: a modified version of ConvLSTM with added residual connections from input to output of the LSTM cell.

Model	1 step	5 step	10 step	Objects Lost
RCNet (Oh et al., 2015)	0.0061	0.0140	0.0268	1.0
ConvLSTM (Xingjian et al., 2015)	0.0026	0.0303	0.0503	0.4
ConvLSTM + Residual	0.0026	0.0141	0.0210	0.45
SpatialNet	**0.0024**	**0.0114**	**0.0176**	**0.13**

Table 1. MSE for multi-step prediction on PhysVideos (lower is better). All models were trained with 1 step prediction loss. SpatialNet suffers least from compound errors during prediction, and is able to maintain objects and dynamics more consistently (Figure 3). Number of objects lost (after 20 steps) was determined manually by evaluating 15 random videos in the test set.

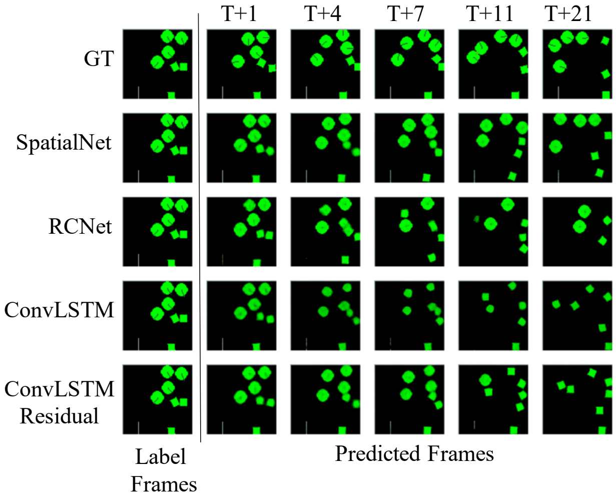

Figure 3. Visualization of multi-step predictions of SpatialNet, RCNet, and ConvLSTM variants, along with ground truth (GT). After 20 steps of self prediction, SpatialNet maintains the internal wall and all seven objects in the scene while RCNet (Oh et al., 2015) loses the internal wall and 3 of the moving objects. ConvLSTM loses shape information and has less accurate dynamics prediction. SpatialNet is most consistent in obeying physical laws.

We train all prediction models using mean squared error (MSE) loss. We use the Adam optimizer (Kingma and Ba, 2015) in our experiments with a learning rate of 10^{-4}.

Results From Table 1, we see that SpatialNet achieves a lower test MSE compared to all the baselines, especially for multi-step predictions. This suggests that SpatialNet encourages better dynamic generalization compared to RCNet and ConvLSTM. We can also observe from Figure 3, that SpatialNet is able to accurately maintain the number of objects in the video even after 20 steps, while the baselines suffer from merging of objects (RCNet) or loss of shape information (ConvLSTM). Further, SpatialNet is also able to maintain background details such as walls that are quickly lost in RCNet, as the spatial memory structure allows the input to easily remember fixed background information. We also find that the spatial memory's overall structure allows it to be very resistant to input noise as well as better general-

Model	Drag	Elasticity
SpatialNet (random init)	35.8	43.8
SpatialNet (PT on Atari Pong)	35.0	33.6
ConvLSTM (PT on PhysVideos)	57.2	53.2
SpatialNet (PT on PhysVideos)	69.8	56.9
SpatialNet (full train)	78.5	67.8

Table 2. Accuracies on predicting drag and elasticity from video frames (PT = pre-training)

ize to unseen environments – please see the supplementary material for detailed analyses.

4.2 Predicting physical parameters

To further probe the representations learned by the frame prediction models, we test their ability to predict physical properties of environments (e.g. elasticity or drag) from videos. We train a 2 layer classification model on top of the hidden state representations produced by SpatialNet/ConvLSTM to predict one of 3 values for elasticity/drag - low, medium or high. Only the classification layers are trained, while the rest of the parameters are kept fixed (except for *full train*).

From Table 2, we see that randomly initialized parameters or SpatialNet trained on Atari Pong don't do well, indicating that they don't capture physics. SpatialNet trained on PhysVideos gets an accuracy of around 69% on drag prediction (close to the fully trained model accuracy of 78%). This shows that the pre-training indeed helps the model acquire priors over physical dynamics. Further, the low numbers of the model trained on Atari Pong indicate that task-specific frame prediction may not generalize well.

4.3 Reinforcement Learning

In this section, we describe the use of SpatialNet to accelerate reinforcement learning. We first train SpatialNet on the physics video dataset described in the previous section. Then, we use the pre-trained SpatialNet model as a future frame predictor for a reinforcement learning agent. We perform empirical evaluations on three different platforms - a suite of 2D games (PhysWorld), a 3D partially observable environment, and a stochastic version of Atari games (Machado et al., 2017a). We demonstrate that IPA with SpatialNet pre-training outperforms existing approaches in all platforms. The IPA architecture also allows for effective decoupling of environment dynamics from agent policy, which results in better transfer performance across tasks.

Experimental setup In our experiments, we use SpatialNet to predict the next k[†] future frames. We then stack the current frame with the k predicted frames and use this as input to a model free policy. We use the Adam optimizer with learning rate 10^{-4} to train model predictions and the same set of hyper-parameters for training all policy agents as those used in (Schulman et al., 2017). For our policy network, we use the architecture described in (Mnih et al., 2015). We report numbers averaged over 3 different random seeds.

Baselines We compare our agent (IPA) with a number of different baselines:

1. *PPO*: A standard implementation of Proximal Policy Gradient (PPO) (Schulman et al., 2017), which is model-free and uses the current frame with the last k frames to output an action. The number of frames provided to PPO is the same as that provided to IPA.

2. *PPO + VF*: PPO with value function expansion (Feinberg et al., 2018), which uses a dynamics predictor to obtain a more consistent estimate of the current state's value.

3. *I2A*: Imagination Augmented Agent (Weber et al., 2017) uses a combination of past frames and a recurrent encoding of future rollouts[‡] as input to the policy.

4. *ISP*: A variant of IPA that uses the hidden layer of SpatialNet directly as input to a policy network.

5. *JISP* : ISP with auxiliary frame prediction loss.

6. *Other frame predictors*: Finally, we also consider baselines where we augment our agent, IPA, with future frames predicted by RCNet (Oh et al., 2015) and ConvLSTM (Xingjian et al., 2015).

PhysWorld We first consider PhysWorld, a new collection of three physics-centric 2D games that we created. These games require an agent to accurately predict object movements and rotations in order to perform well. All three tasks have an environment consisting of around 10 randomly moving boxes and circles as well as up to three internal impassable walls. *PhysGoal* is a navigation task to reach goals while avoiding objects, *PhysForage* is an object gathering task, and *PhysShooter* requires a stationary agent to shoot selected moving objects while preventing collisions. The objects in each of these environments are *different colors and sizes* than those used to train the dynamics predictor in Section 4. We provide a detailed description of each task in the supplementary material. We emphasize that the main parameters (like object velocities, rotations,etc.) in the PhysWorld games are fully **randomized** for each episode. To obtain good performance, agents need a good understanding of general physics and cannot just memorize frames.

[†] We find k=3 to work well in our experiments.

[‡] Rollouts are k future frames predicted by SpatialNet.

	PhysGoal	*PhysForage*	*PhysShooter*
PPO	17.9 (0.8)	44.2 (5.4)	23.2 (1.2)
PPO + VF	19.2 (2.4)	40.4 (5.4)	26.1 (2.9)
I2A + SpatialNet (action-cond)	4.2 (0.4)	23.7 (3.1)	16.5 (1.8)
I2A + SpatialNet	16.4 (6.2)	20.8 (2.0)	19.3 (0.7)
IPA + RCNet	20.7 (3.1)	46.3 (23.4)	31.7 (1.0)
IPA + ConvLSTM	21.6 (2.1)	39.5 (7.0)	29.1 (1.6)
ISP	15.2 (1.2)	45.3 (5.5)	18.6 (1.1)
JISP	18.2 (5.5)	**124.3** (27.1)	28.6 (1.5)
IPA + SpatialNet (Blink)	24.6 (2.8)	48.5 (5.3)	31.0 (1.9)
IPA + SpatialNet (PhysVideos)	**30.8** (5.2)	50.6 (11.5)	**42.3** (2.9)

Table 3. Average scores (with standard deviation) obtained in PhysWorld environments by various agents after 10 million frames of training. Scores are rewards over 100 episodes, averaged over runs with 3 different random seeds. IPA + SpatialNet consistently outperforms the other approaches. RCNet, SpatialNet, ConvLSTM are pretrained on PhysVideos. PPO+VF = PPO with Value Function Expansion. SpatialNet (Blink) refers to a model trained on videos with blinking objects. We add 500K additional frames to the PPO baselines to account for the frames used in pre-training for the other models.

Results: We detail the performance of our approach compared to the baselines in Table 3 and show learning curves in Figure 4. Quantitatively, we find that our approach, IPA + SpatialNet (PhysVideos), obtains significant gains over most baselines in all three tasks in PhysWorld using IPA with SpatialNet. We find that IPA with RCNet or ConvLSTM provides less benefits, due to slower learning than SpatialNet. We also find PPO with value expansions (PPO+VF) also provides slight gains, but significantly less than the gains conferred by IPA. I2A leads to no gains in performance, since it generates a global encoding of an image, destroying local dynamics information of objects. Both ISP and JISP perform worse than IPA except on PhysForage. On PhysForage, we find that JISP performs better, likely due to increased policy capacity compared to IPA (i.e. more parameters). We observe that SpatialNet trained on videos with blinking objects does not provide as much of a benefit, pointing to the fact that our full model is learning some aspects of dynamics beyond just object appearances.

IPA encourages the policy to take into account the future physics of objects, a bias crucial for good performance on each of the PhysWorld environments. Qualitatively, we observe that in all three environments, IPA agents navigate to goals and collect objects with more confidence, even if there are nearby obstacles nearby. In PhysShooter, IPA agents are much more able to hit objects further away on the map, which require multiple time-steps before collisions. Figure 4 demonstrates how having a good prior results in faster learning of the environment dynamics of PhysShooter.

Figure 4 shows the relative training rates of policies under PPO and IPA. In **Phys-Shooter** we see immediate benefits in using a physics model, as physics knowledge of the future is crucial as the agent only gets one action approximately every 4-5 frames. In **Phys-Goal** and **Phys-Forage**, we see long term benefits in knowing future physics as this knowledge allows the agents to more efficiently collect points.

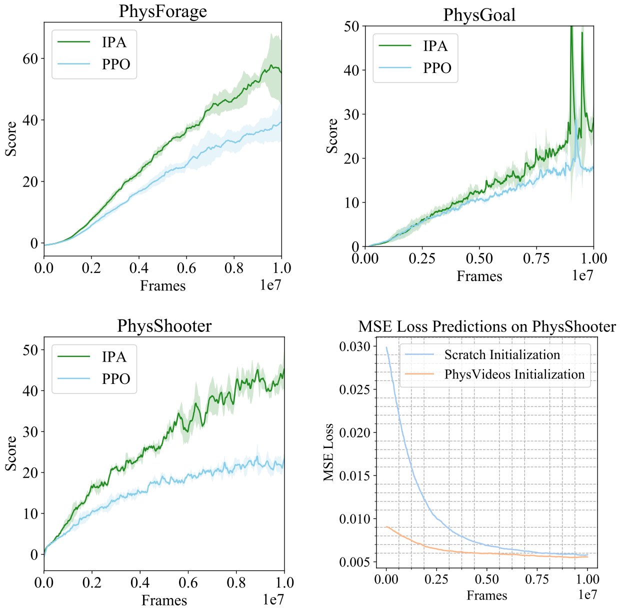

Figure 4. Training curves on PhysWorld and MSE curve (bottom-right) for predicting future frames in PhysShooter.

PhysShooter3D Additionally, we also evaluate on PhysShooter3D, a 3D physics game which we construct using Bullet (Coumans, 2010). We add gravity to the world and generate moving projectiles that follow bouncing parabolic trajectories. We then render 2D images from a particular viewpoint, causing moving objects to be partially or fully occluded at times. With these additional factors, learning dynamics is even more challenging. The game requires a stationary agent to fire bullets at selected 3D projectiles without itself being hit by any projectiles. We found that PPO obtained a score of 0.86 ± 0.28 while IPA + SpatialNet obtained 1.73 ± 0.09 and IPA using Ground truth frames obtained 4.16 ± 0.84. This demonstrates that IPA generalizes well to partially observed settings, with still room for improvement by performing better frame prediction.

Stochastic Atari Games In addition to PhysWorld and PhysWorld3D, we also investigate the performance of IPA on a stochastic version of the Arcade Learning Environment (ALE) (Bellemare et al., 2013), by adding *sticky actions*, where an agent repeats its last action with probability $p = 0.5$. This stochasticity was shown to be the most challenging type of randomization to add to ALE (Hausknecht and Stone, 2015; Machado et al., 2017a). We evaluate performance on all Atari games, a subset of which are shown in Table 4. All Atari experiments are run with 5 different seeds.

	PPO	I2A	IPA
Assault	2932 (153)	**3249.7** (378)	2968.4 (124)
Asteroids	1321 (233.5)	1340 (351)	**2098** (102)
Breakout	19.7 (0.9)	18.7 (0.0)	**23.4** (1.0)
DemonAttack	5510 (412)	5492 (233)	**6793** (558)
Enduro	376.7 (10.5)	380 (8.0)	**398.6** (23.0)
FishingDerby	6.7 (10.1)	**12.1** (4.0)	9.3 (3.0)
Frostbite	1342 (2154)	1649 (2100)	1701 (2485)
IceHockey	-5.9 (0.3)	-6.3 (0.0)	-6.1 (0.0)
Pong	**6.6** (14.1)	-1.4 (15.0)	2.2 (13.0)
Tennis	-6.3 (2.1)	-8 (4.0)	**-3.8** (1.0)

Table 4. Scores (and standard deviation) obtained on Stochastic Atari Environments with *sticky actions* (actions repeated with 50% probability at each step). Scores are average performance over 100 episodes after 10M training frames, over 5 different random seeds with included standard deviations.

We emphasize that this is an *out-of-domain* evaluation – we use the prior trained on PhysVideos to initialize the dynamics predictor for Atari, which contains a significantly different pixel space. Further, not all Atari games are reliant on understanding physics and we do not expect our approach to provide significant gains on those environments.

Results: From Table 4, we observe that IPA outperforms PPO in 8 out of the 10 different tasks[§] – these are all games that contain physical interactions between objects and benefit from our prior. In several games like Enduro, Breakout, Frostbite, FishingDerby and Assault, IPA provides benefits later on in training after the agent has figured out a good initial policy. In others like Asteroids and DemonAttack, IPA shows immediate boosts in training performance, resulting in faster policy learning. On Pong, where IPA performed worse than PPO, we found that the agents learned to place paddles at one particular location where without paddle movement, the ball would bounce and score points. Similarly, on Ice Hockey, where PPO outperformed IPA, we found that agents can learn a repetitive strategy to prolong the game indefinitely, removing the need for tracking dynamics information. Under such situations, there is no added advantages to predicting dynamics, explaining the reduced scores of IPA. We provide additional qualitative results, including frame predictions, in the supplementary material.

4.4 Transfer and Generalization

We now present some empirical results under the transfer scenario and provide some analysis of our model. Table 5 also shows the impact of initializing IPA with different pre-trained dynamics models on the PhysShooter environment. We find that initializing SpatialNet with random parameters does not perform very well, but using a SpatialNet pretrained on PhysVideos provides better performance (see Figure 4 for MSE errors). Moreover, we observe that transferring a SpatialNet model fine-tuned on a different task like PhysForage/PhysGoal results in even greater performance improvements. *Interestingly, we note that transferring just the dynamics model in IPA results in a larger performance gains than transferring both model and policy*. For instance, transferring the model from PhysForage results in a score of 53.7 while transferring both model+policy gets a lower score of 40.4. The former is a 27% increase compared to using just PhysVideos (42.27) , while the latter results in a lower score. This provides further evidence that decoupling model learning from policy learning allows for better generalization.

Source env	Agent	Model transfer	Policy transfer	Reward
None	PPO	-	-	23.2
None	IPA	-	-	35.42
PhysVideos	IPA + SpatialNet	Y	-	42.27
PhysGoal	PPO	-	Y	25.42
	IPA + SpatialNet (Fix)	Y	N	26.30
	IPA + SpatialNet (FT)	Y	N	**42.83**
	IPA + SpatialNet (FT)	Y	Y	42.44
PhysForage	PPO	-	Y	24.47
	IPA + SpatialNet (Fix)	Y	N	30.30
	IPA + SpatialNet (FT)	Y	N	**53.66**
	IPA + SpatialNet (FT)	Y	Y	40.40

Table 5. Effects of model initialization and transfer on training policies in *PhysShooter*. Topmost section shows baseline PPO, random initialization of dynamics for IPA, and pre-trained IPA using PhysVideos. The bottom two sections demonstrate results while transferring different models from two other games – direct policy (PPO), transfer dynamics model and fix it (Fix), transfer dynamics and finetune (FT), and transfer both dynamics+policy and finetune. IPA allows decoupling of policy transfer from model transfer, allowing better transfer in cases of environment similarity but task dissimilarity. Scores obtained on the PhysWorld environments after training for 10M frames and evaluated by taking average rewards of the last 100 training episodes.

5 Conclusion

We have proposed a new approach to model-based reinforcement learning by learning task-agnostic dynamics priors. First, we pre-train a frame prediction model (SpatialNet) on raw videos of a variety of objects in motion. We then use this network to initialize a dynamics model for an RL agent, which makes use of predicted frames as additional context for its policy. Through several experiments on three different domains, we demonstrate that our approach outperforms model-free techniques and approaches that learn environment dynamics from scratch. We also demonstrate the generalizability of our dynamics predictor through transfer learning experiments.

Acknowledgements We would like to thank Alexander Botev, John Schulman, Tejas Kulkarni, Bowen Baker and the OpenAI team for providing helpful comments and sug-

[§] Results on all Atari games are in supplementary material.

gestions.

References

André Barreto, Will Dabney, Rémi Munos, Jonathan J Hunt, Tom Schaul, Hado P van Hasselt, and David Silver. Successor features for transfer in reinforcement learning. In *Advances in neural information processing systems*, pages 4055–4065, 2017.

Peter W. Battaglia, Razvan Pascanu, Matthew Lai, Danilo Rezende, and Koray Kavukcuoglu. Interaction networks for learning about objects, relations and physics. In *NIPS*, 2016.

Marc G Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279, 2013.

Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289*, 2015.

Erwin Coumans. Bullet physics engine. *Open Source Software: http://bulletphysics. org*, 2010.

Mark Cutler and Jonathan P How. Efficient reinforcement learning for robots using informative simulated priors. 2015.

Mark Cutler, Thomas J Walsh, and Jonathan P How. Reinforcement learning with multi-fidelity simulators. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, pages 3888–3895. IEEE, 2014.

Peter Dayan. Improving generalization for temporal difference learning: The successor representation. *Neural Comput.*, 5(4): 613–624, 1993.

Rachit Dubey, Pulkit Agrawal, Deepak Pathak, Thomas L Griffiths, and Alexei A Efros. Investigating human priors for playing video games. *ICML*, 2018.

Vladimir Feinberg, Alvin Wan, Ion Stoica, Michael I Jordan, Joseph E Gonzalez, and Sergey Levine. Model-based value estimation for efficient model-free reinforcement learning. *arXiv preprint arXiv:1803.00101*, 2018.

Chelsea Finn, Ian Goodfellow, and Sergey Levine. Unsupervised learning for physical interaction through video prediction. In *NIPS*, 2016.

Matthew J Hausknecht and Peter Stone. The impact of determinism on learning atari 2600 games. 2015.

Ken Kansky, Tom Silver, David A Mély, Mohamed Eldawy, Miguel Lázaro-Gredilla, Xinghua Lou, Nimrod Dorfman, Szymon Sidor, Scott Phoenix, and Dileep George. Schema networks: Zero-shot transfer with a generative causal model of intuitive physics. In *ICML*, 2017.

Jintao Ke, Hongyu Zheng, Hai Yang, and Xiqun Chen. Short-term forecasting of passenger demand under on-demand ride services: A spatio-temporal deep learning approach. *CoRR*, abs/1706.06279, 2017.

Michał Kempka, Marek Wydmuch, Grzegorz Runc, Jakub Toczek, and Wojciech Jaśkowski. Vizdoom: A doom-based ai research platform for visual reinforcement learning. In *Computational Intelligence and Games (CIG), 2016 IEEE Conference on*, pages 1–8. IEEE, 2016.

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

Tejas D Kulkarni, Ardavan Saeedi, Simanta Gautam, and Samuel J Gershman. Deep successor reinforcement learning. *arXiv:1606.02396*, 2016.

Marlos C. Machado, Marc G. Bellemare, Erik Talvitie, Joel Veness, Matthew J. Hausknecht, and Michael Bowling. Revisiting the arcade learning environment: Evaluation protocols and open problems for general agents. *CoRR*, abs/1709.06009, 2017a. URL http://arxiv.org/abs/1709.06009.

Marlos C Machado, Clemens Rosenbaum, Xiaoxiao Guo, Miao Liu, Gerald Tesauro, and Murray Campbell. Eigenoption discovery through the deep successor representation. *arXiv preprint arXiv:1710.11089*, 2017b.

Michael Mathieu, Camille Couprie, and Yann LeCun. Deep multi-scale video prediction beyond mean square error. *arXiv preprint arXiv:1511.05440*, 2015.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. In *NIPS Workshop*, 2013.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nat.*, 518(7540):529–533, 2015.

Damian Mrowca, Chengxu Zhuang, Elias Wang, Nick Haber, Li F Fei-Fei, Josh Tenenbaum, and Daniel L Yamins. Flexible neural representation for physics prediction. In *Advances in Neural Information Processing Systems*, pages 8799–8810, 2018.

Duy Nguyen-Tuong and Jan Peters. Using model knowledge for learning inverse dynamics. In *ICRA*, pages 2677–2682, 2010.

Alex Nichol, Vicki Pfau, Christopher Hesse, Oleg Klimov, and John Schulman. Gotta learn fast: A new benchmark for generalization in rl. *arXiv preprint arXiv:1804.03720*, 2018.

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.

Emilio Parisotto and Ruslan Salakhutdinov. Neural map: Structured memory for deep reinforcement learning. *CoRR*, abs/1702.08360, 2017.

Deepak Pathak, Pulkit Agrawal, Alexei A Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction. In *ICML*, 2017.

Viorica Patraucean, Ankur Handa, and Roberto Cipolla. Spatio-temporal video autoencoder with differentiable memory. *CoRR*, abs/1511.06309, 2015.

Pymunk. Pymunk. http://www.pymunk.org/en/latest/. Accessed: 2018-09-26.

Jonathan Scholz, Martin Levihn, Charles Isbell, and David Wingate. A physics-based model prior for object-oriented mdps. In *International Conference on Machine Learning*, pages 1089–1097, 2014.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *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, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of go with deep neural networks and tree search. *Nat.*, 529(7587):484–489, 2016.

Richard S Sutton. Integrated architectures for learning, planning, and reacting based on approximating dynamic programming. In *ICML*, 1990.

Valentin Thomas, Jules Pondard, Emmanuel Bengio, Marc Sarfati, Philippe Beaudoin, Marie-Jean Meurs, Joelle Pineau, Doina Precup, and Yoshua Bengio. Independently controllable factors. *arXiv preprint arXiv:1708.01289*, 2017.

Nicholas Watters, Andrea Tacchetti, Theophane Weber, Razvan Pascanu, Peter Battaglia, and Daniel Zoran. Visual interaction networks. In *NIPS*, 2017.

Théophane Weber, Sébastien Racanière, David P Reichert, Lars Buesing, Arthur Guez, Danilo Jimenez Rezende, Adria Puigdomènech Badia, Oriol Vinyals, Nicolas Heess, Yujia Li, et al. Imagination-augmented agents for deep reinforcement learning. *arXiv preprint arXiv:1707.06203*, 2017.

Chris Xie, Sachin Patil, Teodor Moldovan, Sergey Levine, and Pieter Abbeel. Model-based reinforcement learning with parametrized physical models and optimism-driven exploration. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 504–511. IEEE, 2016.

SHI Xingjian, Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai-Kin Wong, and Wang-chun Woo. Convolutional lstm network: A machine learning approach for precipitation nowcasting. In *Advances in neural information processing systems*, pages 802–810, 2015.

Haiyan Yin, Jianda Chen, and Sinno Jialin Pan. Hashing over predicted future frames for informed exploration of deep reinforcement learning. *arXiv preprint arXiv:1707.00524*, 2017.

Amy Zhang, Harsh Satija, and Joelle Pineau. Decoupling dynamics and reward for transfer learning. *arXiv preprint arXiv:1804.10689*, 2018a.

Chiyuan Zhang, Oriol Vinyals, Remi Munos, and Samy Bengio. A study on overfitting in deep reinforcement learning. *arXiv:1804.06893*, 2018b.

Susan Zhang, Michael Petrov, Pachoki Jacob, Henrique Pond, Brooke Chan, Filip Wolski, Szymon Sidor, Rafa Jzefowicz, Przemysaw Dbiak, David Farhi, Greg Brockman, Jonathan Raiman, Jie Tang, Christy Dennison, Paul Christiano, Shariq Hashme, Larissa Schiavo, Ilya Sutskever, Eric Sigler, Jonas Schneider, John Schulman, Christopher Hesse, Jack Clark, Quirin Fischer, Diane Yoon, Christopher Berner, Scott Gray, Alec Radford, and David Luan. Openai five, 2018c.

Guangming Zhu, Liang Zhang, Peiyi Shen, and Juan Song. Multimodal gesture recognition using 3-d convolution and convolutional lstm. *IEEE Access*, 5:4517–4524, 2017.

Guangxiang Zhu, Zhiao Huang, and Chongjie Zhang. Object-oriented dynamics predictor. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 9826–9837. Curran Associates, Inc., 2018. URL http://papers.nips.cc/paper/8187-object-oriented-dynamics-predictor.pdf.