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.

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.

---

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

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.

Formally, consider a standard Markov Decision Process (MDP) setup represented by the tuple $(S, A, T, R)$, 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 $\Omega$, and given the current state $s_t$, we first apply our prediction model iteratively to obtain future state predictions:

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

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

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

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 \(r_t = \frac{\pi_t(a_t | s_t, h_t)}{\pi_{\text{old}}(a_t | s_t, h_t)}\) and the advantage, \(A_t\), is computed using the value function \(V(s_t, \hat{s}_{t+1}, ..., \hat{s}_{t+k})\). 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 \(E\) that converts an input image \(x_t\) into a 3D representational map \(z_t\). The second module is a spatial memory block, \(\sigma\), 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 \(D\) that predicts the next frame \(x_{t+1}\) from \(o_t\). Both the encoder and decoder modules (\(E\) and \(D\)) use two convolutional layers each with residual connections.

We implement the spatial memory block \(\sigma\) 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:

\[
\begin{align*}
  i_t &= f(C_c \oplus [h_t; z_t]); \\
  u_t &= f(C_u \oplus [i_t; h_t]) \\
  h_{t+1} &= f(C_{dyn} \oplus u_t); \\
  o_t &= f(C_d \oplus [z_t; h_{t+1}])
\end{align*}
\]

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 \(D = \{(x_1^{(i)}, x_2^{(i)}, ..., x_T^{(i)})\}_{i=1}^N\), we learn the parameters of the model using a standard MSE-based loss function, \(L(\theta) = \sum_{t} \sum_{j} \|x_j^t - \hat{x}_j^t\|^2\).

SpatialNet is inspired by the ConvLSTM model but is different from ConvLSTM in that while ConvLSTM performs an additive state updation operation \(c_t = f_t \cdot c_{t-1} + i_t \cdot \tanh(W_c x_t + W_h h_{t-1} + b_h)\), 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 (Py- munk). Each video in the dataset has frames of size $84 \times 84 \times 3$ 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.

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-
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$^\dagger$ 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 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.

---

<table>
<thead>
<tr>
<th>Model</th>
<th>Drag</th>
<th>Elasticity</th>
</tr>
</thead>
<tbody>
<tr>
<td>SpatialNet (random init)</td>
<td>35.8</td>
<td>43.8</td>
</tr>
<tr>
<td>SpatialNet (PT on Atari Pong)</td>
<td>35.0</td>
<td>33.6</td>
</tr>
<tr>
<td>ConvLSTM (PT on PhysVideos)</td>
<td>57.2</td>
<td>53.2</td>
</tr>
<tr>
<td>SpatialNet (PT on PhysVideos)</td>
<td>69.8</td>
<td>56.9</td>
</tr>
<tr>
<td>SpatialNet (full train)</td>
<td>78.5</td>
<td>67.8</td>
</tr>
</tbody>
</table>

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

$^\dagger$ Rollouts are $k$ future frames predicted by SpatialNet.
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.

PhysShooter3D Additionally, we also evaluate on PhysShooter3D, a 3D physics game which we construct using Bullet (Coulman, 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.
We find that initializing SpatialNet with random parameters does not perform very well, but using a SpatialNet pretrained on PhysVideos provides better performance (see Table 4).

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 pretrained 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 Table 4).

### 4.4 Transfer and Generalization

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 advantage to predicting dynamics, explaining the reduced scores of IPA. We provide additional qualitative results, including frame predictions, in the supplementary material.

<table>
<thead>
<tr>
<th>Source env</th>
<th>Agent</th>
<th>Model transfer</th>
<th>Policy transfer</th>
<th>Reward</th>
</tr>
</thead>
<tbody>
<tr>
<td>None</td>
<td>PPO</td>
<td>-</td>
<td>-</td>
<td>23.2</td>
</tr>
<tr>
<td>None</td>
<td>IPA</td>
<td>-</td>
<td>-</td>
<td>35.42</td>
</tr>
<tr>
<td>PhysVideos</td>
<td>IPA + SpatialNet</td>
<td>Y</td>
<td>-</td>
<td>42.27</td>
</tr>
<tr>
<td>PhysGoal</td>
<td>IPA + SpatialNet (Fix)</td>
<td>Y</td>
<td>N</td>
<td>26.30</td>
</tr>
<tr>
<td></td>
<td>IPA + SpatialNet (FT)</td>
<td>Y</td>
<td>N</td>
<td>42.83</td>
</tr>
<tr>
<td></td>
<td>IPA + SpatialNet (FT)</td>
<td>Y</td>
<td>Y</td>
<td>42.44</td>
</tr>
<tr>
<td>PhysForage</td>
<td>IPA + SpatialNet (Fix)</td>
<td>Y</td>
<td>N</td>
<td>30.30</td>
</tr>
<tr>
<td></td>
<td>IPA + SpatialNet (FT)</td>
<td>Y</td>
<td>Y</td>
<td>32.14</td>
</tr>
<tr>
<td></td>
<td>IPA + SpatialNet (FT)</td>
<td>Y</td>
<td>Y</td>
<td>40.40</td>
</tr>
</tbody>
</table>

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 fine-tune (FT), and transfer both dynamics+policy and fine-tune. 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.

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.

### 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-
gestions.

References


