TASK-AGNOSTIC DYNAMICS PRIORS FOR DEEP REINFORCEMENT LEARNING

Yilun Du [*]
Department of Computer Science
Massachusetts Institute of Technology
77 Massachusetts Avenue, Cambridge, MA, USA
yilundu@mit.edu

Karthik Narasimhan
Department of Computer Science
Princeton University
110 Morrison Hall Princeton, NJ 08544
karthikn@cs.princeton.edu

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; 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 agents policy and its knowledge of environment dynamics, which can result in heavy overfitting to a particular task Zhang et al. (2018b). 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. One way to alleviate this problem is to initialize dynamics models with universal *task-agnostic* priors to allow for more efficient and stable model-based RL.

One such prior is an understanding of physical laws, which proves quite valuable in navigating the world. 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 policies. In order to obtain a prior for physical dynamics, we perform unsupervised learning over raw videos containing moving physical objects objects. The parameters of the model implicitly capture general laws of physics. For model-based RL, we initialize the dynamics model with these pre-trained parameters and fine-tune them on the specific task, while simultaneously learning a policy for the task. We utilize the dynamics model to predict future frames up to a finite horizon, which are then used as additional input into a policy network. Our use of *task-agnostic* dynamics priors allows for better generalization across environments.

Learning a good future dynamics model is challenging mainly due to: a) isolation of the dynamics of each entity and b) accurate modeling of localized interactions around the entity. To overcome these issues, we propose SpatialNet, with a spatial memory block that 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, generalization and size invariance.

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. (2017). In all scenarios, we first demonstrate the value of learning a task-agnostic prior for model dynamics Further, we also show that the dynamics model fine-tuned on these tasks transfer better to new tasks.

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. The second direction is to incorporate physics priors into parameterized dynamics models for future state prediction.

[*]Work done during fellowship at OpenAI

Video prediction models. Our frame prediction model is closest in spirit to the ConvolutionalLSTM (Xingjian et al., 2015) Similar architectures that incorporate differentiable memory modules have been proposed (Patraucean et al., 2015), 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 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. Incorporating physics priors into learning dynamics models of environments has been the subject of some recent work. Cutler et al. (2014); Cutler and How (2015) learn Bayesian nonparametric priors for physical systems and use it to guide policy search. Scholz et al. (2014) model latent physical parameters like mass and friction and use an external physics simulator to predict future states for planning. (Kansky et al., 2017) learn a generative physics simulator and demonstrate its usefulness in transferring policies across task variations. Xie et al. (2016) develop a model that incorporates prior knowledge about the dynamics of rigidbody systems in order to perform model-predictive control. 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.

3 FRAMEWORK

Our goal is to demonstrate that acquiring task-agnostic dynamics priors from raw videos helps agents learn faster in new environments. We first train a suitable neural network to predict pixels in the next frame given the previous k frames of video. We use videos of objects moving according to classical mechanics, without any extra annotations. Next, we then use the pre-trained frame predictor to initialize the dynamics model for an RL agent. The dynamics model is used to predict a few frames into the future and use them as additional context for a control policy. During this phase, the dynamics model is also simultaneously fine-tuned using trajectories observed in the task environment.

3.1 REINFORCEMENT LEARNING WITH DYNAMICS PREDICTORS

Consider a Markov Decision Process (MDP) setup represented by the tuple $\langle S, A, T, R \rangle$, where S is the set of all possible states, 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:

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

We then train a policy network to output actions using all these predicted states as input in addition to the current state, $a_t = \pi(s_t, \hat{s}_{t+1}, \ ... \ \hat{s}_{t+k})$.

For the policy network, we follow the network in Mnih et al. (2015) and train using Proximal Policy Optimization (PPO) (Schulman et al., 2017) algorithm. Simultaneously, we also update the parameters of the dynamics model using the transitions from the environment. However, policy gradients are not back-propagated to the dynamics predictor. We call this agent an Intuitive Physics Agent (IPA) since it first learns an intuitive prior of physical interactions and uses it to learn policies.

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

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

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 (Diagram in Supplement). The first module is a standard convolutional encoder $\mathcal{E}$ that converts an input image x into a 3D representational map z. The second module is a spatial memory block, σ, that converts z and the hidden state h from the previous timestep into an output representation o and new hidden state h'. Finally, we have a convolutional decoder $\mathcal{D}$ that predicts the next frame x' from o. 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:

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

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)). 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, with convolution as dynamics, while static information such as background texture are passed directly through the input encoding z_t and dynamics is . 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.

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 model physical interactions and predict future states (Section 4.1). 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 $84 \times 84 \times 3$ with 4-8 different shapes (such as squares or circles) moving inside a room with up to 3 randomly generated interior walls.

4.1 Frame Prediction

Baselines We compare our model, SpatialNet, with RCNet (Oh et al., 2015), which uses a LSTM encoder, *ConvLSTM* (Xingjian et al., 2015) which uses all inner operations of an LSTM with convolutions and ConvLSTM + Residual from input encoding to output encoding. 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 Overall, we find that SpatialNet achieves significantly lower test multi-step MSE and fewer loss objects than all other baselines. SpatialNet is able to maintain both objects (which fails on RCNet) and shape (which fails in ConvLSTM). Furthermore, SpatialNet is able to maintain background details. We also find that SpatialNet structure allows it to generalize better to datasets of different dynamics and input sizes, as well as noised inputs. We provide quantitative numbers, and analysis of generalization, as well as qualitative results in the supplement.

Environment	PPO	PPO + VF	IPA + RCNet	IPA + ConvLSTM	I2A + SpatialNet	ISP	JISP	IPA + SpatialNet (Blink)	IPA + SpatialNet (PhysVideos)
PhysGoal	17.7 (0.1)	19.2 (2.4)	20.7 ± 3.1	21.56 ± 2.1	16.4 ± 6.2	15.2 ± 1.2	18.2 ± 5.5	24.6 ± 2.8	**30.8** ± 6.2
PhysForage	38.9 ± 8.9	40.4 ± 5.4	46.3 ± 23.4	39.47 ± 7.0	20.75 ± 2.0	45.3 ± 5.5	**124.3** ± 27.1	48.8 ± 5.3	50.6 ± 11.5
PhysShooter	23.2 ± 1.3	26.1 ± 2.9	31.7 ± 1.0	29.1 ± 1.6	19.3 ± 0.7	18.6 ± 1.1	28.6 ± 1.5	31.0 ± 1.9	**42.3** ± 2.9

Table 1: Average scores (along with standard deviation) obtained in PhysWorld environments 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.

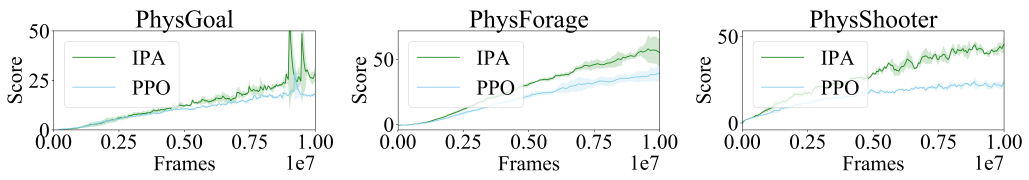

Figure 1: Training curves for PPO and IPA agents on PhysWorld environments. In PhysGoal and PhysForage, IPA demonstrates better performance during later stages of training. In PhysShooter, IPA provides better performance early on because in this game planning is essential since a player can only fire one bullet at a time.

4.2 Predicting physical parameters

To further probe the representations learned by the frame prediction models, we also tested their ability to predict physical properties of environments (e.g. elasticity or drag) from videos. Overall, we found that SpatialNet learned better physical representations than ConvLSTM. Furthermore, we found that learning from physics videos allowed significantly better physical parameter prediction compared to other domains such as Atari. We provide quantitative numbers and details in the supplement.

4.3 Reinforcement Learning

In this section, we describe the use of SpatialNet to accelerate reinforcement learning 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., 2017). We demonstrate that IPA with SpatialNet pre-training outperforms existing approaches in all platforms.

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 1e-4 to train model predictions and the same set of hyper-parameters as 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 with number of input frames balanced, PPO (Schulman et al., 2017) , PPO + VF (Feinberg et al., 2018), I2A (Weber et al., 2017), ISP, where we use the hidden layer of SpatialNet as input to a policy network, JISP, where we include auxiliary frame prediction loss with ISP and IPA with other frame prediction models.

PhysWorld We first consider PhysWorld, a new collection of three fully randomized physics-centric 2D games that we created. These games require an agent to accurately predict object movements and rotations in order to perform well. *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. Each environment has *different colors and sizes or objects* than those used to train the dynamics predictor in Section 4. We provide a detailed description of each task in the supplementary material.

Results: We detail the performance of our approach and baselines in Table 1 and show learning curves in Figure 1. 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 gain, I2A leads to no gains

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

Label	Assault	Asteroids	Breakout	DemonAttack	Enduro
PPO	2932.2 ± 153.2	1321.0 ± 233.5	19.7 ± 0.9	5510.9 ± 412.5	376.7 ± 10.5
IPA	**2968.4** ± 124.0	**2098.4** ± 102.0	**23.4** ± 1.0	**6793.6** ± 558.0	**398.6** ± 23.0
MSE	0.0023	0.0023	0.00029	0.0032	0.00230
Label	**FishingDerby**	**Frostbite**	**IceHockey**	**Pong**	**Tennis**
PPO	6.7 ± 10.1	1342.5 ± 2154.5	**-5.9** ± 0.3	**6.6** ± 14.1	-6.5 ± 2.1
IPA	**9.3** ± 3.0	**1701.1** ± 2485.0	-6.1 ± 0.0	2.2 ± 13.0	**-3.8** ± 1.0
MSE	0.00150	0.00110	0.000035	0.00016	0.00075

Table 2: 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.

in performance, likely due to a global encoding of a 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 capacity compared to IPA. 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.

Figure 1 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 (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). The original ALE is fully deterministic (except for random starts) and hence, a dynamics predictor would not provide much value. We modify ALE by adding *sticky actions*, where an agent repeats its last action with probability $p = 0.5$. We evaluate performance on all Atari games and in more detail on selected subset of 10 games though we thought had relevant physics features before evaluation. Experiments on all Atari environments were run with 5 seeds.

Results: From Table 2, we observe that IPA outperforms PPO in 8 out of the 10 different tasks on pre-selected physics based games, and outperforms PPO on 31 of the 49 total evaluated Atari games. 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, we found that agents can learn to permanently stall the ball. Under such situations, there is no added advantages to predicting dynamics, explaining the reduced scores of IPA. Interestingly, we found that MSE error of prediction in Atari is significantly lower than in the physics environments, indicating determinism still in Atari.

4.4 Transfer and Generalization

We further analyze transfer of IPA across different tasks. We find that initializing SpatialNet with random parameters does not perform very well, but using a pretrained SpatialNet pretrained on PhysVideos provides better performance. 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. This provides further evidence that decoupling model learning from policy learning allows for better generalization. We provide a quantitative numbers in the supplement.

5 Conclusion

We have proposed a new approach to model-based reinforcement learning by learning task-agnostic dynamics priors and have demonstrated its advantages over model-free techniques in transfer and performance.

Acknowledgements We would like to thank Alexander Botev, John Schulman, Tejas Kulkarni, Bowen Baker for providing helpful comments and suggestions as well as everyone else at OpenAI.

References

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.

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.

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.

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

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, 2017. URL `http://arxiv.org/abs/1709.06009`.

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.

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.

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.

A Model Overview

We provide an overview of SpatialNet in Figure 2.

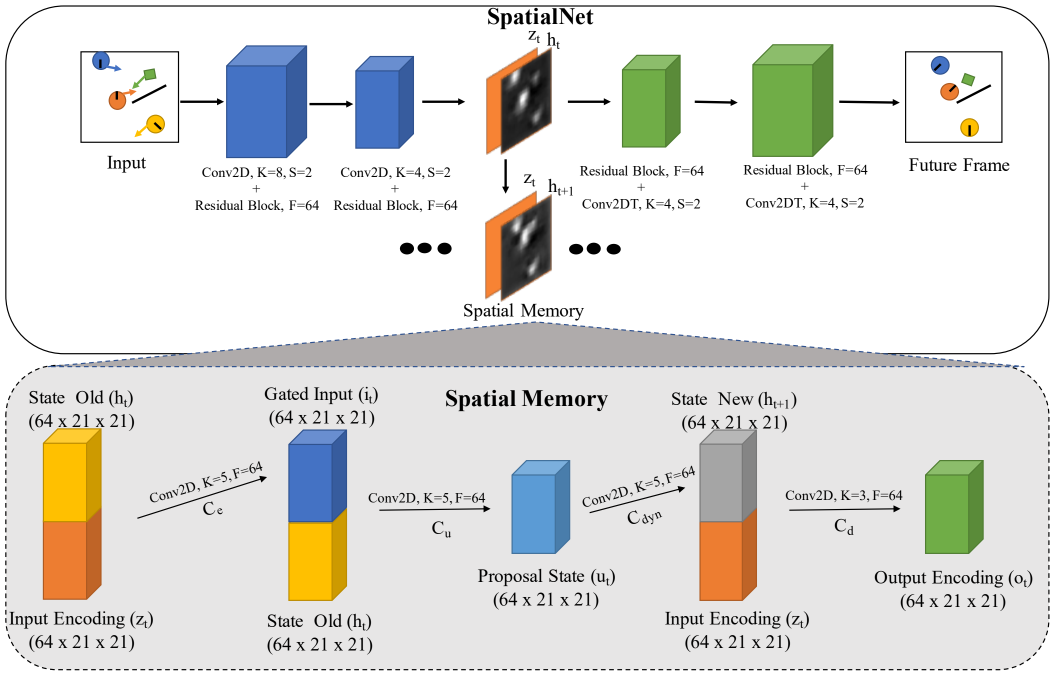
SpatialNet
Input
Conv2D, K=8, S=2 + Residual Block, F=64
Conv2D, K=4, S=2 + Residual Block, F=64
z_t h_t
z_t h_{t+1}
Residual Block, F=64 + Conv2DT, K=4, S=2
Residual Block, F=64 + Conv2DT, K=4, S=2
Future Frame
Spatial Memory
State Old (h_t) (64 x 21 x 21)
Input Encoding (z_t) (64 x 21 x 21)
Conv2D, K=5, F=64 C_e
Gated Input (i_t) (64 x 21 x 21)
State Old (h_t) (64 x 21 x 21)
Conv2D, K=5, F=64 C_u
Proposal State (u_t) (64 x 21 x 21)
Conv2D, K=5, F=64 C_{dyn}
State New (h_{t+1}) (64 x 21 x 21)
Input Encoding (z_t) (64 x 21 x 21)
Conv2D, K=3, F=64 C_d
Output Encoding (o_t) (64 x 21 x 21)

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.

B Additional Dynamic Prediction Experiments

B.1 Frame Prediction

Model	1 step	5 step	10 step	Objects Lost
RCNet	0.0061	0.0140	0.0268	1.0
ConvLSTM	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 3: 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.

From Table 3, 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. Trajectories in the supplement show that SpatialNet is able to accurately maintain the number of objects in 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 spatial memory's overall structure allows it to be very resistant to input noise as well as better generalize to unseen environments – please see the supplementary material for detailed analyses.

ϵ	RCNet	ConvLSTM	SpatialNet (ours)
0	0.0061	0.0026	**0.0024**
0.1	0.0078	0.0030	**0.0026**
0.5	0.0268	0.0072	**0.0062**

Table 4: MSE loss on physics prediction data-set on on single-step prediction with test inputs corrupted with Gaussian noise of magnitude ϵ (model trained with no corruption). Due to its local nature, SpatialNet suffers less form errors in inputs and is able to maintain object numbers/dynamics more consistently even with domain shift.

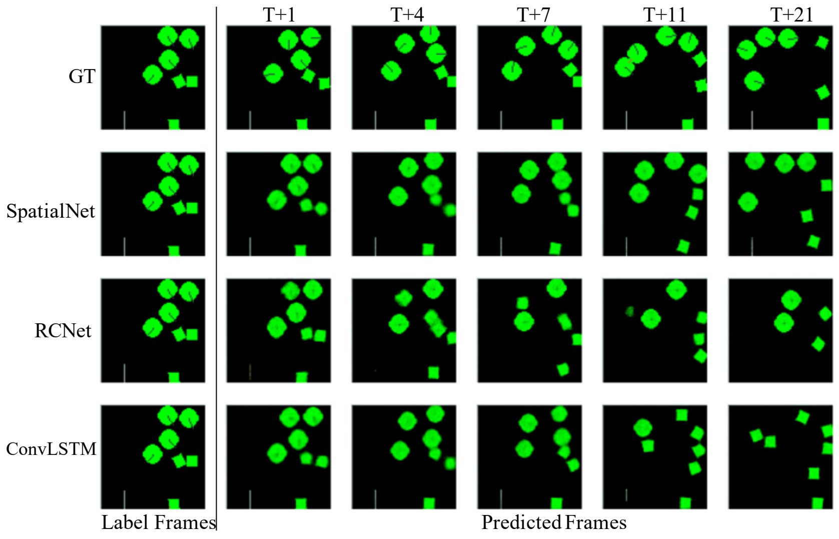

Figure 3: Visualization of multi-step predictions of SpatialNet, RCNet, and ConvLSTM variants, along with ground truth (GT). After 20 steps of self prediction, SpatialNet maintains the internal wall and all seven objects in the scene while RCNet loses the internal wall and 3 of the moving objects. ConvLSTM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

B.2 Sensitivity to Corruption of Inputs

We investigate the effects of noisy observations in the input domain at test time on both SpatialNet and RCNet, by adding different amounts of Gaussian random noise to input images (Table 4). We find that SpatialNet is more resistant to noise addition. SpatialNet predictions are primarily local, preventing compounding of error from corrupted pixels elsewhere in the image whereas RCNet compresses all pixels into a latent space, where small errors can easily escalate.

B.3 Qualitative Visualization

We provide visualizations of multistep rollout predictions from models in Figure 3. We provide visualizations of video prediction on each of the generalization datasets in Figure 4 and Figure 5.

B.4 Dataset Generalization.

We test generalization by evaluating on two unseen datasets. For the first, we create a test set where objects are half the size of the training set and initialized randomly with approximately twice the starting velocity. In this new regime, we found that RCNet had a MSE of 0.0115, ConvLSTM has a MSE of 0.0067, while SpatialNet had a MSE of 0.0039. We find RCNet is unable to maintain shapes of the smaller objects, sometimes omitting them, while ConvLSTM maintains shape but is unable to adapt to new dynamics. In contrast, SpatialNet local structure allows it to generate new shapes, and its dynamic seperation allows better generalization. In the second dataset, we explore input size invariance. We create a second testing data-set consisting 16-32 random circles and squares and input images of size 168x168x3 (the density of objects per area is conserved). On this dataset, we obtained a MSE of 0.0042 compared to ConvLSTM of 0.0060, which is comparable to the MSE on the original

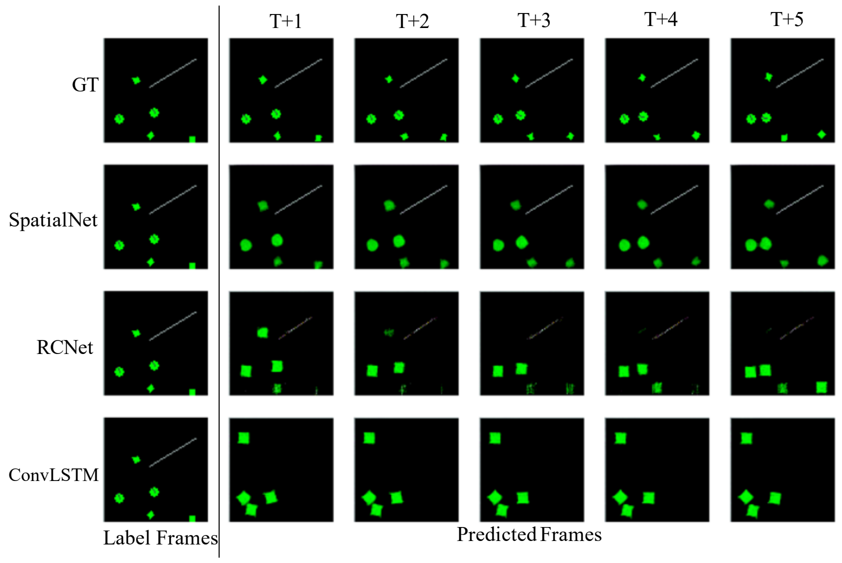

Figure 4: Predictions of SpatialNet, RCNet on test data-set with objects twice as small and with twice the movement speed as trained on. All shown frames are one step predictions. SpatialNet is able to accurately generalize to smaller, faster objects while RCNet is unable to generate the shapes of the smaller objects and suffers from background degradation and ConvLSTM is unable to maintain shapes and dynamics.

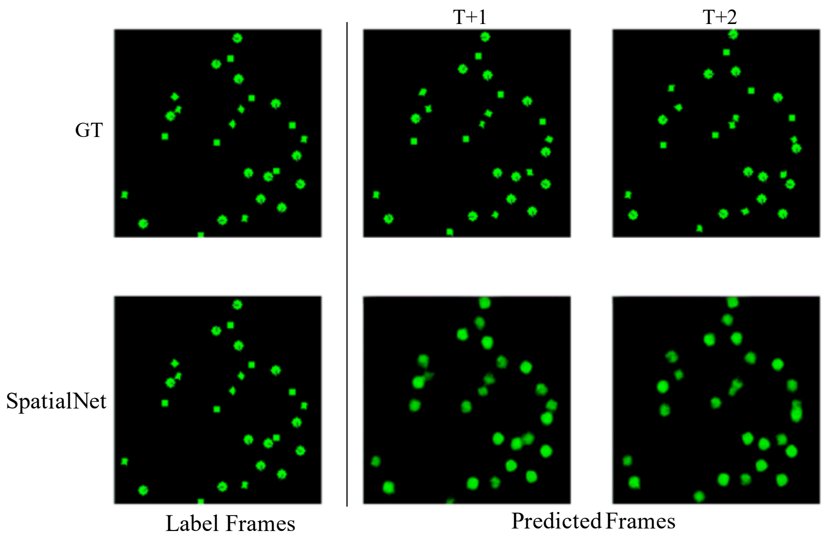

Figure 5: Predictions of SpatialNet on input images of 168 x 168 when SpatialNet was trained on 84 x 84 images. Prediction shown are 1 step future predictions. SpatialNet is able to maintain physical consistency in at large input sizes.

test dataset of 0.0024, showing that the spatial memories local structure allows to easily generalize to different input image sizes. We show qualitative plots of both datasets in the supplementary.

B.5 Physical Parameter Prediction

We trained a 2 layer classification model on top of hidden state representations produced by SpatialNet/ConvLSTM to predict one of 3 values for elasticity/drag - low, medium or high on set of physical videos. Only the classification layers are trained, while the rest of the parameters are kept fixed (except for *full train*).

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 5: Accuracies on predicting drag and elasticity from video frames (PT = pre-training)

From Table 5, we see that randomly initialized parameters or SpatialNet trained on Atari Pong dont do well, indicating that they dont 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.

C PhysWorld

All environments consisting of around 10 randomly moving boxes and circles as well as up to three internal impassable walls. We provide a description of the three games environments in PhysWorld:

PhysGoal: In this environment, an agent has to navigate to a large red goal. Each successful navigation (+1 reward) respawns the red goal at a random location while collision with balls or boxes terminates the episode (-1 reward).

PhysForage: Here, an agent has to collect moving balls while avoiding moving boxes. Each collected ball (+1 reward) will randomly respawn at a new location with a new velocity. Collision with boxes lead to termination of episode (-1 reward).

PhysShooter: In PhysShooter, the agent is stationary and has to choose an angle to shoot bullets. Each bullet travels through the environment until it hits a square (+1 reward) or circle (-1 reward) or leaves the screen. If a moving ball or box hits the agent (-1 reward), the episode is terminated. After firing a bullet, the agent cannot fire again until the bullet disappears.

Examples of agents playing the PhysWorld environments are given in Figure 6.

C.1 Plots of Dynamics Learning

We show plots of dynamics learning with or without a learned physics prior in Figure 8

C.2 SpatialNet Predictions

Figure 7 shows the qualitative next 3 frame predictions of SpatialNet on each of the different PhysWorld environment with the first frame being the current observation. In PhysGoal, SpatialNet is able to infer the movement of the obstacles, the dark blue agent, and the red goal after agent collection. In PhysGather, SpatialNet is able to infer movement of obstacles as well as the gather of a circle. In PhysShooter, SpatialNet is able to anticipate a collision of the bullet with a moving obstacle and further infer the shooting of a green bullet by the agent.

C.3 Visualization of Spatial Memory

We provide visualization of the values of spatial memory hidden state while predicting future frames. We visualize the values of spatial memory on PhysVideos, PhysGoal and the Atari environment Demon Attack in Figure 9. To visualize, we take the mean across the channels of each grid pixel in the spatial memory hidden state. We find strong correspondence between high activation regions in the spatial memory and dynamic objects in the associated ground label of the dynamic objects. We further find that static background, such as walls in the input, goals and platforms appear to be passed along in input features.

D Additional Atari experiments

We provide plots of training curves on all Atari environments in Figure 10 on provide on quantitative numbers in Figure 6.

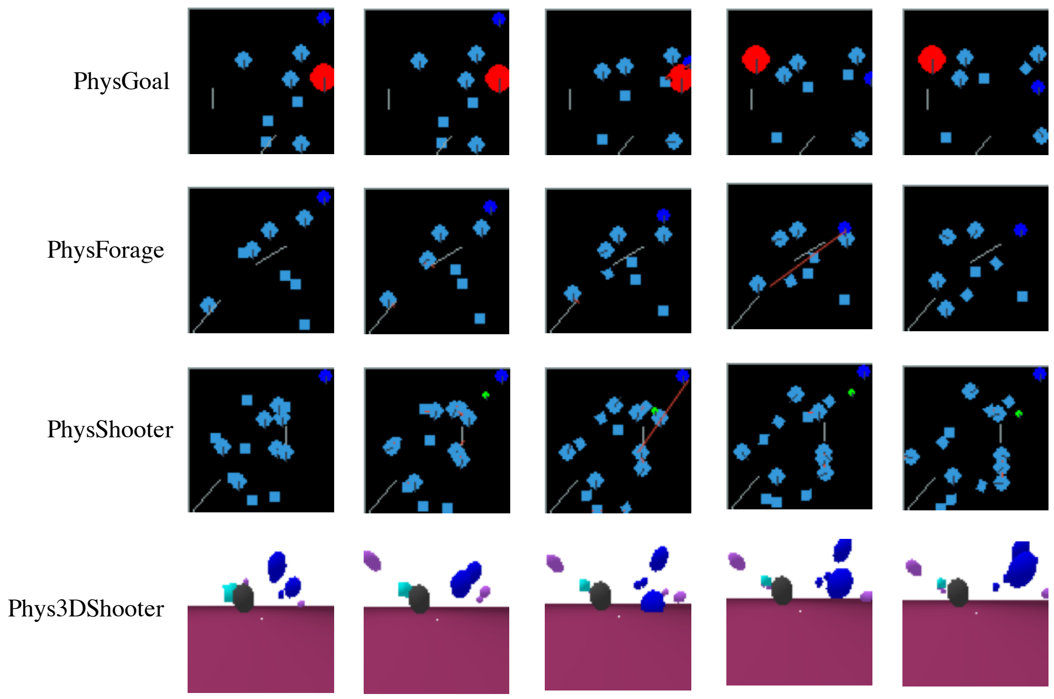

Figure 6: Example agent game-play in each of the PhysWorld environments. In PhysGoal the dark blue agent attempts to reach a red goal while avoiding moving objects. In PhysForage the dark blue agent attempts to gather light blue circles while avoiding squares. In PhysShooter, the dark blue agent is immobile and chooses to fire bullet a green bullet at squares while avoiding circles. In Phys3DShooter, the grey fires turqoise bullets at purple spheres while avoiding blue spheres.

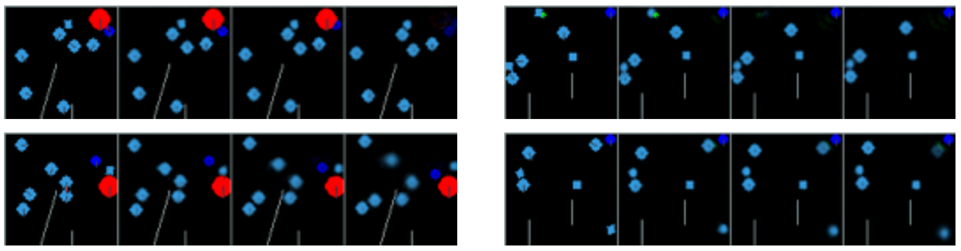

Figure 7: Future image prediction on PhysGoal (left) and PhysShooter (right). First image is current observation, the next three are predicted. SpatialNet is able to predict future dynamics of boxes and balls and anticipate agent movement (PhysGoal) and agent shooting (PhysShooter).

Predictions on Atari We also investigate the benefits (in terms of MSE) of initializing SpatialNet pretrained on the physics dataset compared to training with scratch in Figure 7. We evaluate the MSE error at 1 million frames and find that initializing with the physics dataset provides a 12.9% decrease in MSE error. We find that pretraining helps on 7 of the 10 Atari environments, with the most negatively impacted environment being Enduro, a 3D racecar environment in which the environmental prior encoded by the physics dataset may be detrimental. More significant gains in transfer may be achievable by using a large online database of 2D YouTube videos which cover even more of diversity of games.

SpatialNet Predictions We further visualize qualitative results on SpatialNet on training Atari in Figure 11. In general, across the Atari Suite, we found that SpatialNet is able to accurately model both the environment and agents behavior. In the figure, we seed that SpatialNet is able to accurately predict agent movement and ice block movement in Frostbite. On DemonAttack, SpatialNet is able to infer the falling of bullets. On Asteroid, SpatialNet is able to infer the movement of asteroids. Finally, on FishingDerby, SpatialNet is able to the right player capturing a fish and also predict that the left player is likely to catch a fish (indicated by the blurriness of the rod). We note that any blurriness

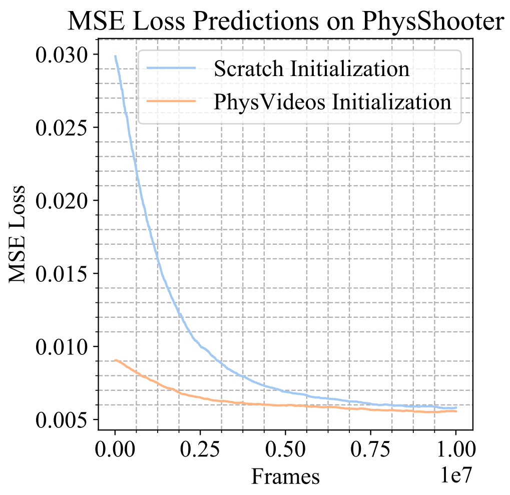

Figure 8: Learning curves demonstrating the impact of using priors learned from PhysVideos on PhysShooter. We observe significantly faster convergence to lower MSE.

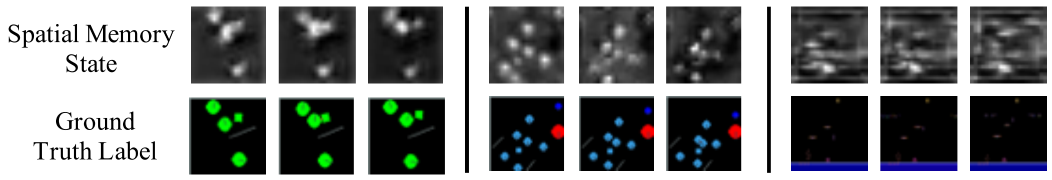

Figure 9: Visualization of SpatialNet hidden state on PhysVideos (left), PhysGoal (middle) and Atari DemonAttack (right). Hidden state has high activations for moving objects while background objects such as walls (left), red goals (middle) and platforms (right) are not attended to as much.

in predicted output may in fact even be beneficial to the policy, since policy can learn to interpret the input. We provide training curves and additional analysis on effects of physics transfer on these environments in the supplementary material.

E Transfer and Generalization

We provide a table showing transfer performance onto PhysShooter from other environments in Table 8. We find that initializing SpatialNet with random parameters does not perform very well, but using a pretrained SpatialNet pretrained on PhysVideos provides better performance. 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. This provides further evidence that decoupling model learning from policy learning allows for better generalization. We provide a quantitative numbers in the supplement.

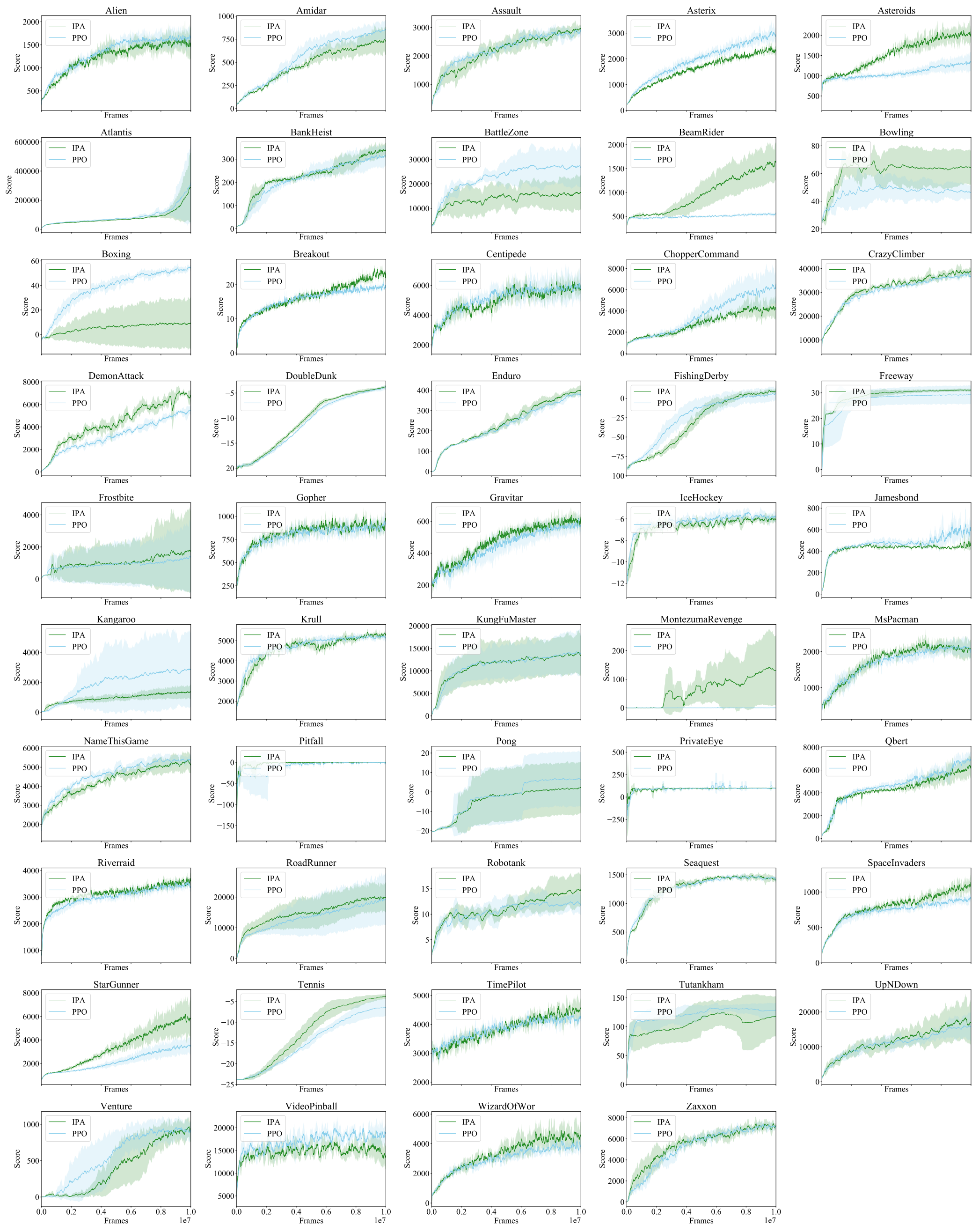

Figure 10: Plots of policy performance trained with either PPO or IPA on all Atari environments on 5 different seeds. IPA sometimes leads to low learning early on the training due to rapid change of 3 predicted future frames. However, later on in training in many different environments, IPA provides performance gains by giving policies future trajectories. IPA provides improvements on 31 of 49 evaluated Atari environments

Environment	PPO	D2A
Alien	**1668.6** ± 224.3	1485.5 ± 281.0
Amidar	**855.9** ± 98.6	725.5 ± 135.0
Assault	2939.2 ± 153.2	**2968.4** ± 124.0
Asterix	**2920.8** ± 287.3	2334.0 ± 184.0
Asteroids	1321.0 ± 233.5	**2098.4** ± 102.0
Atlantis	**323205.4** ± 277643.2	289369.8 ± 239469.0
BankHeist	310.4 ± 44.0	**334.3** ± 29.0
BattleZone	**26828.0** ± 8472.0	16526.7 ± 6986.0
BeamRider	553.1 ± 28.4	**1630.3** ± 400.0
Bowling	46.6 ± 5.2	**64.3** ± 13.0
Boxing	**54.3** ± 2.5	8.9 ± 20.0
Breakout	19.7 ± 0.9	**23.4** ± 1.0
Centipede	**6043.7** ± 990.6	6032.5 ± 199.0
ChopperCommand	**6549.4** ± 1779.1	4112.0 ± 1024.0
CrazyClimber	36893.2 ± 463.9	**38499.0** ± 1221.0
DemonAttack	5510.9 ± 412.5	**6793.6** ± 558.0
DoubleDunk	-4.0 ± 0.5	**-3.8** ± 0.0
Enduro	376.7 ± 10.5	**398.6** ± 23.0
FishingDerby	6.7 ± 10.1	**9.3** ± 3.0
Freeway	29.2 ± 3.6	**31.2** ± 1.0
Frostbite	1342.5 ± 2154.5	**1701.1** ± 2485.0
Gopher	904.0 ± 42.3	**941.1** ± 56.0
Gravitar	574.9 ± 36.2	**627.2** ± 25.0
IceHockey	**-5.9** ± 0.3	-6.1 ± 0.0
Jamesbond	**598.9** ± 112.1	454.3 ± 34.0
Kangaroo	**2842.4** ± 2461.2	1373.0 ± 445.0
Krull	5178.9 ± 205.1	**5219.3** ± 129.0
KungFuMaster	**13831.6** ± 4483.6	13358.5 ± 4352.0
MontezumaRevenge	0.0 ± 0.0	**129.7** ± 122.0
MsPacman	1990.1 ± 227.9	**2097.3** ± 259.0
NameThisGame	**5406.4** ± 278.0	5131.3 ± 427.0
Pitfall	-0.1 ± 0.3	**0.0** ± 0.0
Pong	**6.6** ± 14.1	2.2 ± 13.0
PrivateEye	95.6 ± 5.4	**99.6** ± 0.0
Qbert	**6981.0** ± 548.0	6331.4 ± 769.0
Riverraid	3411.0 ± 201.9	**3612.4** ± 130.0
RoadRunner	19329.6 ± 8472.6	**20041.8** ± 4906.0
Robotank	11.9 ± 1.8	**14.9** ± 3.0
Seaquest	**1426.0** ± 43.5	1408.7 ± 51.0
SpaceInvaders	902.4 ± 66.0	**1132.6** ± 101.0
StarGunner	3450.0 ± 801.5	**5778.5** ± 1584.0
Tennis	-6.5 ± 2.1	**-3.8** ± 1.0
TimePilot	4281.8 ± 126.6	**4580.0** ± 314.0
Tutankham	**128.5** ± 12.3	118.2 ± 35.0
UpNDown	15872.3 ± 3995.3	**16913.7** ± 6344.0
Venture	930.2 ± 137.9	**946.7** ± 167.0
VideoPinball	**18878.1** ± 1251.7	13981.2 ± 2136.0
WizardOfWor	3835.6 ± 404.7	**4629.8** ± 662.0
Zaxxon	7197.4 ± 220.6	**7271.0** ± 264.0

Table 6: Scores 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. IPA helps in 31 out of 49 evaluated Atari games.

Environment	MSE PD	MSE DN	Percent Advantage
Assault	0.00477	0.00522	9.4%
Asteroids	0.002506	0.002518	4.7%
Breakout	0.000417	0.000423	1.4%
DemonAttack	0.00433	0.00562	29.8%
Enduro	0.00576	0.00411	-28.7%
FishingDerby	0.00183	0.00192	4.9%
Frostbite	0.000965	0.00107	10.8%
IceHockey	0.000614	0.0013	111.7%
Pong	0.00636	0.00584	-8.2%
Tennis	0.00142	0.00132	-7.1%

Table 7: MSE on Stochastic Atari Environments (a action is repeated with a geometric distribution with p=0.5) at 1 million training frames. MSE PD is trained with a model from physics dataset while MSE DN is trained with a model from scratch. We evaluate percentage advantage for initializing with a physics dataset as compared to from scratch. We average 12.9% decrease in MSE error using a initialization from pretraining on a physics dataset. The most negative environment, Enduro, involves a 3D landspace which initializing from model trained on a physics data set may be detrimental.

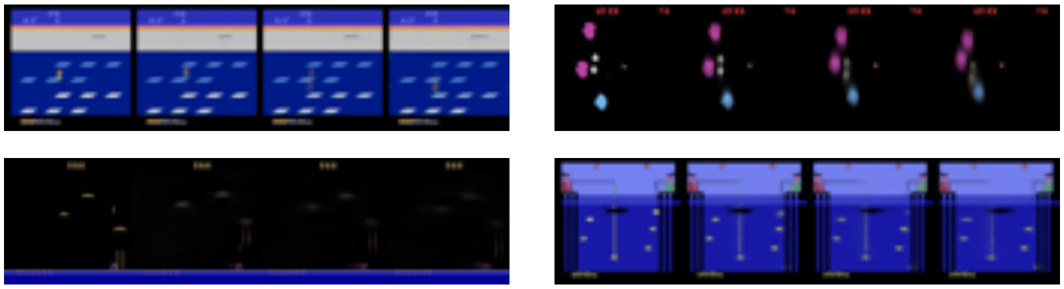

Figure 11: Visualization of model future state prediction on 4 games in Atari (Frostbite - upper left, DemonAttack - lower left, Asteroids - upper right, FishingDerby - lower right). SpatialNet is able to predict falling of bullets, the catching of fish, movement of asteroids, and the movement of tiles/future agent movement in different environments. First frame visualized is ground truth observation, next 3 frames are model future frame predictions.

Source env	What is transferred?	Reward	MSE
None	PPO	23.2	-
None	IPA	35.42	0.00578
PhysVideos	IPA	42.27	0.00554
PhysGoal	PPO	25.42	-
	Fixed SpatialNet	26.30	-
	Finetune SpatialNet	**42.83**	0.00540
	Model + Policy	42.44	0.00540
PhysForage	PPO	24.47	-
	Fixed SpatialNet	30.30	-
	Finetune SpatialNet	**53.66**	**0.00533**
	Model + Policy	40.40	0.00533

Table 8: 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 (Fixed SpatialNet), transfer dynamics and finetune (Finetune SpatialNet), and transfer both dynamics+policy and finetune (Model+Policy). 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.