
Generative Adversarial Networks for Model Based Reinforcement Learning with Tree Search

Ted Xiao*

1st Year M.S.
Machine Learning at Berkeley
Berkeley, CA 94720
ted@ml.berkeley.edu

Gautham Kesineni*

4th Year B.S.
Machine Learning at Berkeley
Berkeley, CA 94720
gautham@ml.berkeley.edu

Abstract

Recent advances in Generative Adversarial Networks (GANs) have shown much promise in the domain of image generation, especially using deep multi-scale architectures. Tree-search methods, on the other hand, have been successful in offline domains but not online learning. This work utilizes a GAN to learn a dynamics model, which is then used for online tree search. We propose a model-based reinforcement learning framework leveraging a combination of video prediction with GANs and online tree-search methods. Our method attempts to combine the near-horizon benefits of a learned dynamics model with the far-horizon benefits of a value network. Experimental results show that the proposed architecture is generally able to produce visually-realistic predictions and achieve performance results somewhat comparable to state-of-the-art such as DQN.

1 Introduction

Over the past several years, the field of deep reinforcement learning (RL) has seen much success in its application of model free approaches to solving many complex planning tasks [15]. The core idea behind model free reinforcement learning is that instead of searching all possible futures of a given state for the action trajectory which best maximizes some objective (as in model based RL), we implicitly learn a mapping from state-action pairs to some measure of expected future value, and then choose an action which maximizes this expected future value. Although the model free approach provably yields a solution which is equivalent to the model based approach, there is an immense computational cost in learning such a value function, which implicitly codes an objective-centric model of the environment with which an agent interacts.

In contrast, model based reinforcement learning often yields immediately optimal agents, with the caveat that the agent has direct access to a model of its environment. However, deep learning approaches to model based RL have been surprisingly less successful, despite the prospect of one-shot optimality. For the most part, applications of deep learning have failed because generative models have been unable to avoid the propagation of uncertainty of state in highly under-constrained Markov Decision Processes (MDPs). With the recent success of generative adversarial approaches for a variety of multi-modal generation tasks, it is natural to wonder if an application of Generative Adversarial Networks (GANs) [8] to model based RL will eliminate the distribution mismatch problem of model learning using neural networks. Furthermore, it is of interest to consider a joint approach to deep RL using the state of the art in model-free approaches in combination with the aforementioned use of GANs.

By using a combination of model-based and model-free approaches, our method allows an agent to plan in any environment by learning a model of the system. We assess the performance of our

*The authors contributed equally to this work

agent in a variety of Atari environments to compare with other state-of-the-art RL algorithms [15]. Although we believe our method should perform well in 3D and real-world environments, Atari provides a baseline to judge our performance on complex yet lower-dimensional problems. Extending our algorithm to 3D and real-world environments is noted later in our paper as an area for further exploration but due to time and resource constraints, we did not explore this ourselves.

2 Related Work and Comparison

Unsupervised learning of video representations for frame prediction is a heavily researched problem in deep learning. A recurrent temporal restricted Boltzmann machine (RTRBM) [5] introduced recurrent connections in RBM to find temporal correlations from sequential data. The structured RTRBM (sRTRBM) extended this to find further structures between hidden variables and observations. However, more recently, Ranzato et al., inspired from language modeling, defined a recurrent architecture for predicting frames in a discrete space of patch clusters [3]. This was later adapted to an LSTM model [4] by Srivastava et al. to generate videos of hand-written digits moving up and down. Guo et al. [7] used an action-conditional model similar to variational-auto-encoders (VAE) to predict future frames in classic Atari games. Mathieu et al. [1] used a multi-scale GAN approach with different losses to generate video for both Atari and real-world data. In contrast to previous works with the exception of Guo et al., we are solving a problem where control variables have a direct effect on temporal dynamics. From our initial tests, we found that recursively applying action-conditional prediction in the VAE architecture proposed by Guo et al. led to further blurriness in the predictions as we expected. By using GANs to learn the dynamics, we hope to alleviate this problem of recursively increasing uncertainty.

In addition, there has been some amount of investigation into the reinforcement learning aspects of this work. In particular, tree search has been applied with success to model-based deep learning applied to Atari games by Guo et al. in [2]. However, that work focuses more on using imitation learning to train on trajectories observed from an agent using Monte Carlo Tree Search. In particular, the agent is an Upper Confidence Bound applied to Tree Search (UCT) agent [13]. The game emulator is used for the model dynamics that allow the state tree to be constructed, but this constrains the agent to act in offline settings only, as the UCT agent is very computationally expensive. Once UCT rollouts are recorded, a new agent that is able to act online is trained using those trajectories. Apart from this work, there has also been interest in learning dynamics models using image observations. Many image to image methods utilizing deep learning have been attempted, including Variational Autoencoders (VAEs) and Convolutional Neural Networks (CNNs). Notably, approaches by N. Wahlstrom et al. and M. Watter et al [9], [11]. attempt to learn dynamics in the latent space. While this work will focus on utilizing multi-scale GANs for image to image model learning, these different approaches can be viewed as a black box. They can all be easily extended or swapped in for our approach.

3 Proposed Approach

There are two primary components to our architecture: action-conditional prediction and tree search. The goal of action-conditional prediction is to learn a function $f : \mathbf{x}_{1:t}, \mathbf{a}_{1:t} \rightarrow \mathbf{x}_{t+1}$, where $\mathbf{x}_{1:t}$ and $\mathbf{a}_{1:t}$ are the frames and actions from time 1 to t . The goal of tree search, assuming a deterministic environment, is to find a sequence of actions that will lead to the highest reward $\sum_{i=1}^t \mathbf{r}_i$. To do this, we recursively apply the learned dynamics function f , substituting \mathbf{x}_t with a generated frame and iterating through all possible actions for \mathbf{a}_t recursively in a tree-like fashion. At the leaves in the tree, defined as nodes at a set terminal depth, we evaluate the image using a value network $v : \mathbf{x}_t \rightarrow \mathbb{E}[z_t | x_t = x, a_{t..T} \sim p]$ that predicts the outcome from position s of games played by using a good policy π , in this case a DQN.

The intuition of our approach is that the learned dynamics model will allow our agent to plan in the near-horizon, while the learned value network will allow our agent to leverage the benefits of far-horizon values for each state. This online approach is vaguely similar to Model Predictive Control (MPC), where the optimal actions for a fixed horizon is computed, but only the first action is actually used.

Algorithm 1: Online Tree Search with Learned Dynamics from GANs

Result: Model-free reinforcement learning guided with learned GAN dynamics

```
1  $\mathcal{D} \leftarrow (s, a, s', r)$  training rollouts on randomized actions;  
2  $\mathcal{Q} \leftarrow$  Action-value function initialized with random weights;  
3  $\mathcal{G} \leftarrow$  Generative dynamics function initialized with random weights;  
4  $\mathcal{D} \leftarrow$  Discriminator dynamics function initialized with random weights;  
5 while optimize do  
6   Root  $r = \text{Node}(s)$ ;  
7   Child queue  $C \leftarrow r$  ;  
8   while  $C$  not empty do  
9     for  $a' \in$  action space of  $s$  do  
10       $s' \leftarrow$  copy of current state  $s$ ;  
11       $s' = \mathcal{G}(s', a')$ ;  
12      if current depth < maximum depth  $d$  then  
13         $C \leftarrow \text{Node}(s')$ ;  
14      else  
15        if  $\mathcal{V}(s') >$  best value then  
16          best value =  $\mathcal{V}(s')$ ;  
17          best node =  $\text{Node}(s')$ ;  
18        end  
19      end  
20    end  
21  end  
22  return first action from  $r \rightarrow$  best node  
23 end
```

3.1 Action-conditional Prediction

The multi-scale network proposed by Mathieu et al. [1] is very similar to what we used, with the addition of more layers to condition on both actions $\mathbf{a}_{t-i:t}$ and frames $\mathbf{x}_{t-j:t}$ where i and j are fixed parameters. While training DQN to create a policy network as stated above, we collect $\mathbf{a}_{1:t}$ and $\mathbf{x}_{1:t}$ to train our action-conditional GAN according to $f : \mathbf{x}_{t-i:t}, \mathbf{a}_{t-j:t} \rightarrow \mathbf{x}_{t+1}$.

We solve many of the traditional problems associated with networks for video prediction, namely (1) convolutions only account for short range dependencies and (2) using an l_2 or l_1 loss produces blurry predictions. These would be very problematic in our approach because (1) would create predictions that are accurate locally but not globally and (2) would lead to compounding uncertainty that would make tree search impossible. (2) is the primary problem we encountered when using Guo et al. [7] for video prediction. As expected, we did not encounter either when using the multi-scale network.

Our multi-scale network is defined as follows. Let u_k be the upscaling factor up to size s_k . X_k^t, Y_k^t are downscaled versions of X^t and Y^t of size s_k where X^t denotes the frames used to generate Y^t , the prediction. a^t is the action given at time step t that would produce Y^{t+1} when run in the simulator. G_k is a component of our network that learns to predict $Y_k - u_k(Y_{k-1})$ from X_k , a guess of Y_k , and a^t . In essence, each component of our network is defined as:

$$\hat{Y}_k = G_k(X) = u_k(\hat{Y}_{k-1}) + G'_k(X_k, u_k(\hat{Y}_{k-1}), a^t) \quad (1)$$

We use these components recursively to make predictions at an increasingly large scale. Our generator network is illustrated in Figure 1 with appropriate modifications for conditioning on actions.

Our discriminator model D predicts the probability whether a frame \hat{Y}^t given frames X^t and actions a^t is the real Y^t or made by our generator G . Note that any Y (and X respectively) is composed of j (and i respectively) frames. Like our generator, our D is also a multi-scale network but with a single output, a probability. We use a modification of the discriminator network proposed by Mathieu et. al. in [1]:

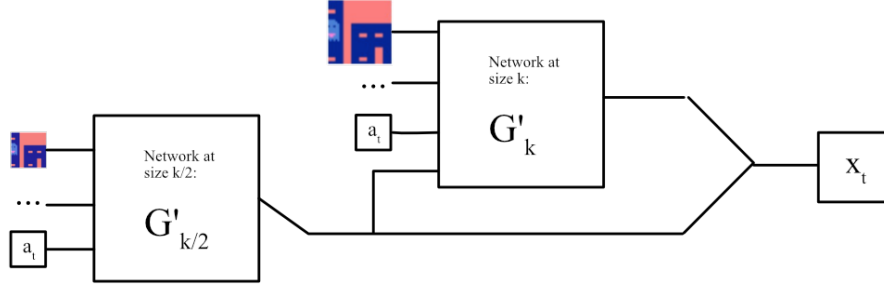


Figure 1: Multi-scale architecture with action-conditioning

$$\mathcal{L}_{adv}^D(X, Y) = \sum_{k=1}^{N_{scales}} L_{bce}(D_k(X_k, Y_k, a^t), 1) + L_{bce}(D_k(X_k, G_k(X), a^t), 0) \quad (2)$$

where L_{bce} is just binary cross-entropy loss, D_k is our discriminator at scale k . The loss for our generator is thus:

$$\mathcal{L}_{adv}^G(X, Y) = \sum_{k=1}^{N_{scales}} L_{bce}(D_k(X_k, G_k(X), a^t), 1) \quad (3)$$

Minimizing this loss is equivalent to increasing the loss of the D , making it harder to discriminate real and generated results. In practice, minimizing these two losses together often leads to instability. This can happen if the generator produces results that sufficiently trick the discriminator without looking like real data. To avoid this problem, we train the generator to minimize a function $\alpha \mathcal{L}_{adv}^G + \beta \mathcal{L}_2$. Increasing β leads to greater stability but also more blurriness. This is a trade-off we've tuned in our network.

3.2 Tree Search

Tree search methods have been commonly used in game theoretic settings. Popular methods include mini-max game trees, alpha-beta pruning, Monte Carlo Tree Search, and B* search [14]. In our approach, we focus on vanilla game tree search in order to demonstrate the effectiveness of online tree search. We discuss possible extensions in the conclusion.

Each game tree is constructed with the current observation as the root of the game tree. For each possible action a' from the current state s , we utilize the learned dynamics to predict the child nodes: $s'_i = \mathcal{G}(s, a')$, $\forall i \in n$, where n is the number of possible actions from current state s . Then, depending on the hyperparameter of terminal depth d , we keep expanding the child nodes using the learned dynamics. Once terminal depth is reached, we calculate the estimated value at each leaf node: $\mathcal{V}(s'_i) = \max_{a'} \mathcal{G}(s'_i, a')$. Given the child with the largest value $\mathcal{V}(s'_i)$, we take the action from the root that led to that child. For an example of what a search tree would look like, see Figure 2.

For the purposes of demonstrating this method in this work, we use a hyperparameter of $d = 1$. It is likely that increasing depth d may result in faster convergence at the cost of exponentially more computation.

4 Experiments

4.1 Processing Data

Our images are normalized observations from OpenAI Gym environments of size 210 x 160 pixels. We used the 'v3' deterministic versions (ex. BreakoutDeterministic-v3) of these games to reduce the randomness from our actions. The v0 environments replayed the previous actions with a probability

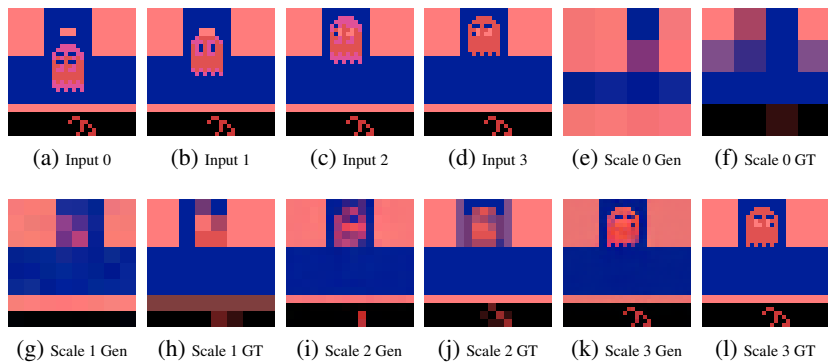
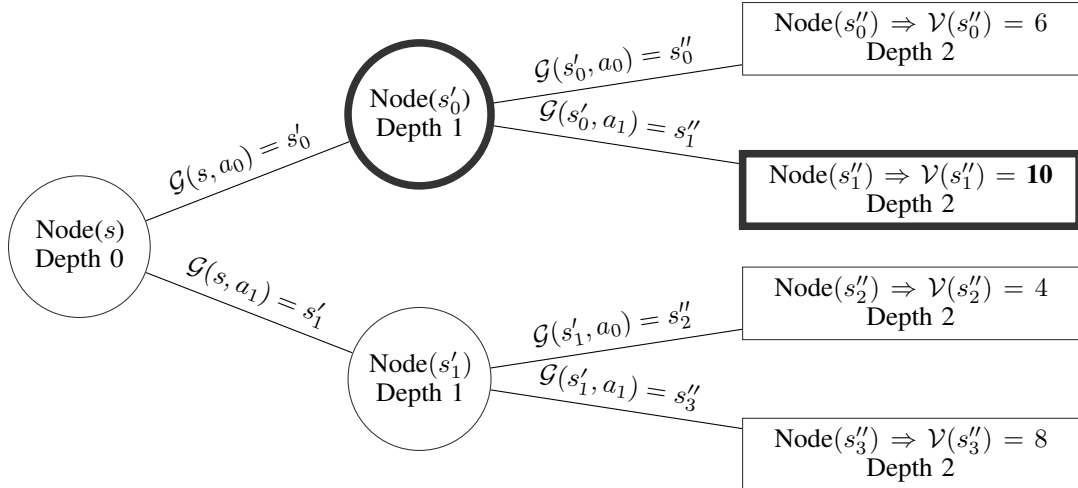


Figure 3: These pictures were made by our generator at each scale given Input frames 0 to 4. Notice that the output can differ at lower scales but produce very accurate predications at higher scales. This is because we sum L_{bce} across all scales for our loss function.

of 0.25 and the non-deterministic environments took a random number of time steps sampled from $\{2, 3, 4\}$ between each state which made it difficult for our action-conditional GAN to converge. We normalized our inputs by mapping the range $[0, 255]$ of pixels to $[-1, 1]$.

We one-hot encode our actions and stretch/reshape them to fit to each scale of the generator. This block of actions is then inserted as another layer to each scale of the network in both the generator and discriminator as is common practice for conditional GANs [12]. We found our model converged well when exposed to 2 or more actions, setting $j \geq 2$. Otherwise, the networks would learn to just ignore the actions given as input. Experimenting with the Atari environments in OpenAI Gym, we found this is likely because it usually takes two actions in any direction to create movement in that direction. Two actions in opposite directions will cancel out and cause no movement on the screen.

4.2 Quantitative Evaluations

In this section, we evaluate both the visual quality of our predictions and the performance of our tree search in various Atari environments.

The images generated by our action-conditional framework can be seen in both Figure 3 and Figure 8. To quantitatively evaluate the quality of our predictions, we used the Peak Signal to Noise Ratio

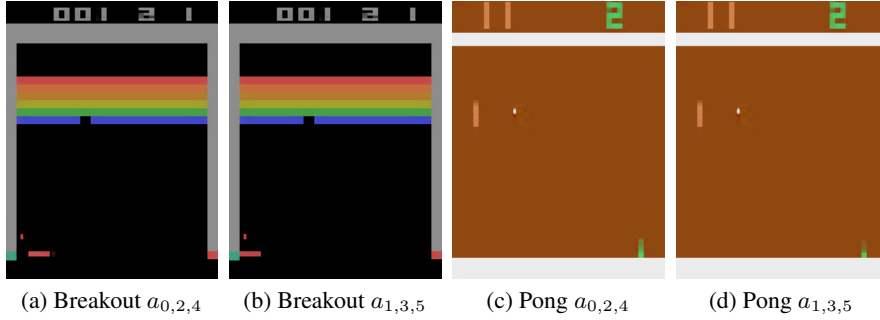


Figure 4: We show the action-conditional output of our network on Pong and Breakout environments. Both have an action space of size 6 but all actions reduce to stopping or continuing movement in direction of motion.

(PSNR), Equation 4, between the true frame Y and our prediction \hat{Y} as well as Sharp. Diff., Equation 5, to measure loss of sharpness from Y to \hat{Y} .

$$\text{PSNR}(Y, \hat{Y}) = 10 \log_{10} \frac{\max_{\hat{Y}}^2}{\frac{1}{N} \sum_{i=0}^N (Y_i - \hat{Y}_i)^2} \quad (4)$$

$$\text{Sharp.Diff.}(Y, \hat{Y}) = 10 \log_{10} \frac{\max_{\hat{Y}}^2}{\frac{1}{N} (\sum_i \sum_j |(\nabla_i Y + \nabla_j Y) - (\nabla_i \hat{Y} + \nabla_j \hat{Y})|)} \quad (5)$$

where $\nabla_i Y = |Y_{i,j} - Y_{i-1,j}|$ and $\nabla_j Y = |Y_{i,j} - Y_{i,j-1}|$.

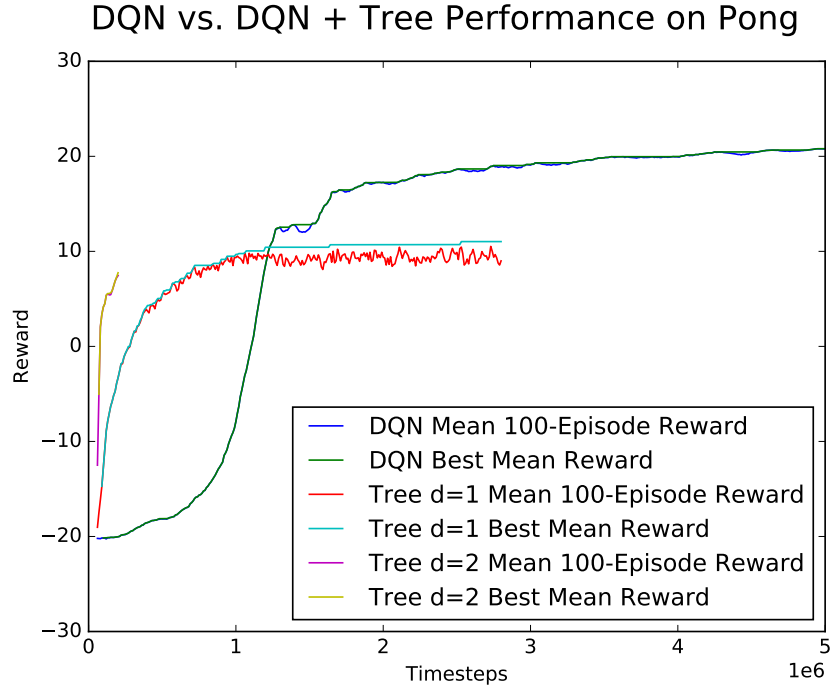
We compared our results for PSNR and Sharp. Diff. with that given by Mathieu et al. [1] as our architecture for generation is rather similar to theirs. We expected some degradation due to the additional task of conditioning on the actions. Thus, the results found in Table 1 were not surprising.

	PSNR	Sharp. Diff.
Breakout non-action	22.9	13.3
Breakout action	14.5	12.8
Pong non-action	22.5	17.2
Pong action	13.7	15.0
Ms.Pacman non-action	26.9	14.5
Ms.Pacman action	23.2	7.9
BeamRider non-action	3.0	12.3
BeamRider action	1.4	10.2

Table 1: PSNR and Sharp. Diff. values given by our action conditional network and the non-action conditional network proposed by Mathieu et al. [1] that represents the state-of-the-art in the space. Each network is trained to 300,000 iterations before the PSNR and Sharp. Diff. values are recorded.

5 Conclusion

This paper introduced a new architecture for model-based reinforcement learning using both model-free and model-based components. We adapt a state-of-the-art multi-scale video prediction algorithm to condition on action as well as previous frames in order learn the dynamics of the game environment. We then use these learned dynamics to perform recursive planning via tree search. Our experimental results have shown there is much promise in this architecture, as we have achieved performance in some environments comparable to that of DQN. Furthermore, since our architecture is domain invariant, we expect it would generalize well to many other visual RL problems including real-world data and perhaps even 3D environments. Since there are many approaches to learning dynamics



(a) Results for Pong

Figure 5: Our method speeds up training initially for Pong but converges at a lower solved rate than the DQN. The tree of depth $d = 2$ took too long to run, but performed well. This suggests that further optimization is required. The results do suggest that an online search tree is potentially powerful in initial scenarios where exploration is crucial in unclear states.

and evaluation game trees, the proposed approach is extendable to a variety of different approaches. To name a few, the dynamics can be learned instead through a VAE, and more complex game trees methods can be utilized, such as Monte Carlo Tree Search. In future work, we hope to study and improve our algorithm’s performance in a larger variety of environments.

Lessons Learned

This project was far harder than we originally imagined. Although we wanted to test a wide variety of environments and algorithms for comparison in our original proposal, we spent so much time building and evaluating our model to its current state that we didn’t have time for further comparisons. Instead, we’ve left that as something we intend to follow up on in the future. The biggest lesson we learned from this project is the importance of planning specific deadlines while creating a proposal and of keeping track with those deadlines over time.

We’ve also learned a great deal from the papers we had to read on our topic. We both learned about the latest in image generation and deep reinforcement learning. This is not our first deep learning graduate class but this class, above any other, has taught us how to carry a research project from beginning to end. A big part of that was how many times we had to pivot our initial hypotheses. This work initially began as an a simple idea of adding Generative Networks to Google’s work on Auxiliary Tasks for Model-free Reinforcement Learning. However, we soon ran into challenges, and pivoted after thinking about model-based methods acting as a guide for model-free learning approaches. Numerous conversations with graduate students and professors helped us along while we were pivoting, and is something we have learned to leverage much more throughout the process.

Acknowledgements

We thank Sergey Levine for fruitful discussions, along with John Schulman, Chelsea Finn, Dawn Song, and Trevor Darrell for organizing the class that made this project possible. In addition, we would like to thank members of the Hybrid Systems Laboratory, including Claire Tomlin, Somil Bansal, and Jaime Fisac, for insightful advice.

References

- [1] M. Mathieu, C. Couprie, & Y. LeCun. Deep multi-scale video prediction beyond mean square error. In *ICLR*, pages 1–14, 2016.
- [2] Xiaoxiao Guo, Satinder Singh, Honglak Lee, Richard L Lewis, and Xiaoshi Wang. Deep learning for real-time atari game play using offline monte-carlo tree search planning. In *Advances in neural information processing systems*, pages 3338–3346, 2014.
- [3] Ranzato, Marc’Aurelio, Szlam, Arthur, Bruna, Joan, Mathieu, Michael, Collobert, Ronan, and Chopra, Sumit. Video (language) modeling: a baseline for generative models of natural videos. *CoRR*, abs/1412.6604, 2014.
- [4] Srivastava, Nitish, Mansimov, Elman, and Salakhutdinov, Ruslan. Unsupervised learning of video representations using LSTMs. In *ICML*, 2015.
- [5] I. Sutskever, G. E. Hinton, and G. W. Taylor. The recurrent temporal restricted Boltzmann machine. In *NIPS*, 2009.
- [6] R. Mittelman, B. Kuipers, S. Savarese, and H. Lee. Structured recurrent temporal restricted Boltzmann machines. In *ICML*, 2014.
- [7] Junhyuk Oh, Xiaoxiao Guo, Honglak Lee, Richard L Lewis, and Satinder Singh. Action-conditional video prediction using deep networks in atari games. In *Advances in Neural Information Processing Systems*, pp. 2863–2871, 2015.
- [8] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *NIPS*, 2014.
- [9] N. Wahlstrom, T. B. Schon, and M. P. Deisenroth, “Learning deep dynamical models from image pixels,” *IFAC-PapersOnLine*, vol. 48, no. 28, pp. 1059–1064, 2015.
- [10] Watter, M., Springenberg, J., Boedecker, J., Riedmiller, M.: Embed to control: A locally linear latent dynamics model for control from raw images. In: *Advances in Neural Information Processing Systems*. pp. 2746–2754 (2015)
- [11] Kevin Sebastian Luck, Gerhard Neumann, Erik Berger, Jan Peters, and Heni Ben Amor. Latent Space Policy Search for Robotics. In *Proc. of IEEE International Workshop on Intelligent Robots and Systems (IROS)*, pages 1434–1440. IEEE, 2014.
- [12] M. Mirza and S. Osindero. Conditional generative adversarial nets. arXiv preprint arXiv:1411.1784, 2014.
- [13] L. Kocsis and C. Szepesvári. Bandit based Monte-Carlo Planning. In *Proc. of European Conference on Machine Learning (ECML)*, pages 283–293. Springer, 2014.
- [14] A. Elnaggar, M. Aziem, M. Gadallah, H. El-Deeb. A Comparative Study of Game Tree Searching Methods. In *International Journal of Advanced Computer Science and Applications (IJACSA)*. Vol. 5, No. 5, 2014.
- [15] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller. Playing atari with deep reinforcement learning. arXiv preprint arXiv:1312.5602, 2013.

6 Appendix

6.1 Failure Cases

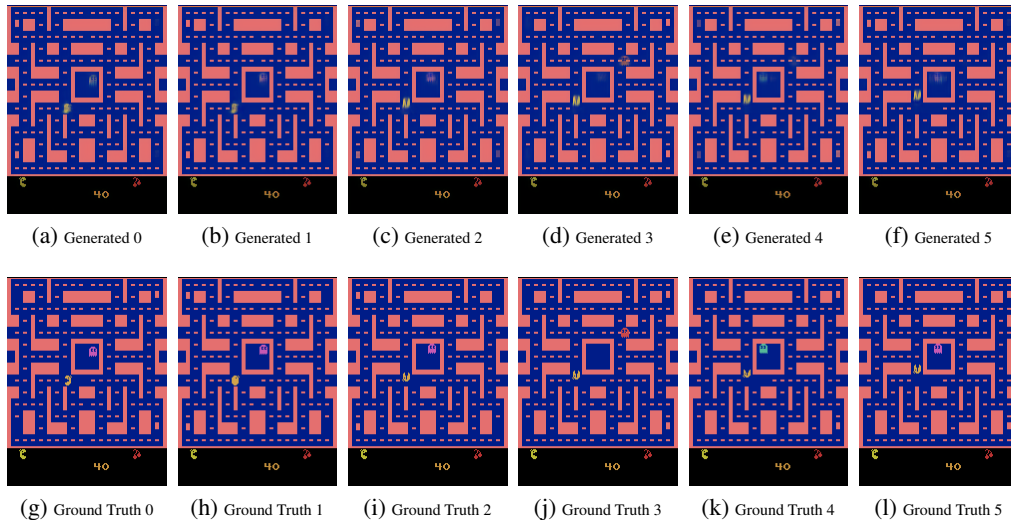


Figure 6: Our action-conditional GAN generating images for the Ms.Pacman environment. These did not make it into the paper but were a central part of our midpoint and final presentations. When combining action-conditional prediction with our tree search approach, we couldn't get Ms.Pacman to converge despite careful parameter tuning. We believe this is because of the large amount of uncertainty on the actions of the ghosts but have yet to verify.

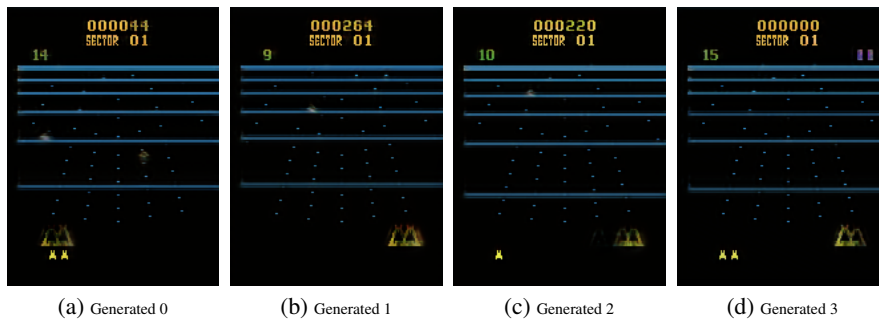
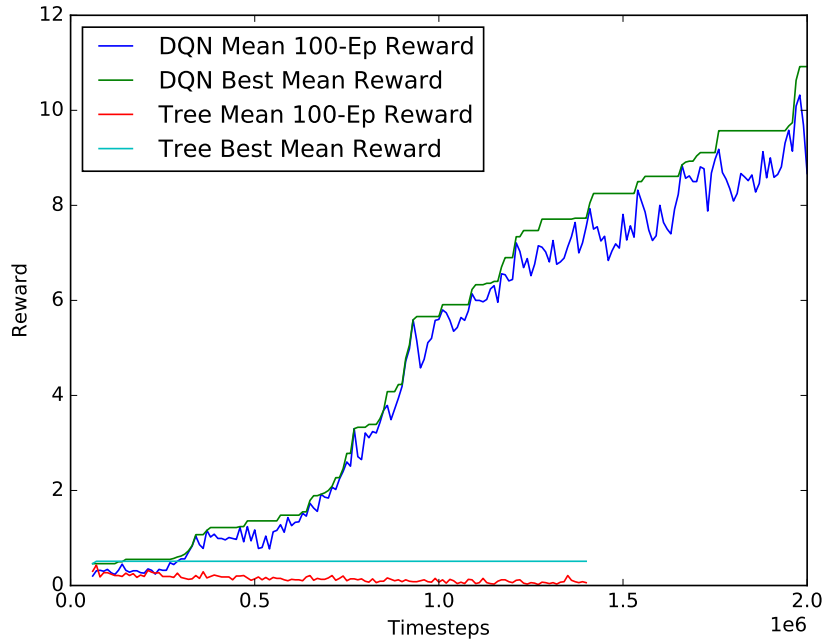


Figure 7: Beam Rider represents a failure case for our prediction algorithm. Even when removing the action-conditional input, our network still failed to converge to a good solution. This is represented by the PSNR and Sharp. Diff. scores in Table 1 which are significantly lower than for other environments. We believe this is because our data collection function samples for areas of movement in the image. Because the entire image is moving, it is thus difficult to focus on a region to collect data from.

6.2 Notes

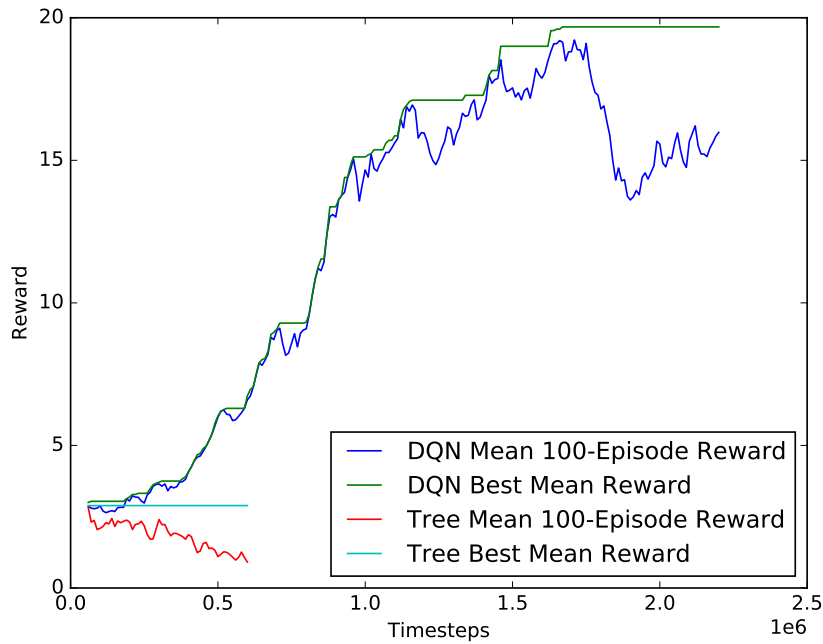
The architecture for our multi-scale video prediction algorithm varies in the number of layers at each level but we did not believe this to be significant enough to include in the paper. Because we condition on actions in addition to frames, we add an additional 2 convolutions and layers to each level to allow the propagation of signals from the actions without further loss of information relative to the vanilla architecture. Despite this, we still do suffer from a reduction in Sharp. Diff. and PSNR.

DQN vs. DQN + Tree Performance on Breakout



(a) Our algorithm does not converge for Breakout. This was surprising to us as Pong is a very similar environment where we achieved comparable results. This warrants further investigation but we believe this may be due to our value function being unable to interpret the noisy output of our action-conditional prediction. Further experiments with a joint training approach could resolve this issue.

DQN vs. DQN + Tree Performance on Beamrider



(b) Our algorithm does not converge for Beamrider. This was not surprising to us as we knew predictions for Beamrider were very bad. Further experiments with a joint training approach may also resolve this issue.

Figure 8: Our approach did not converge well on the Breakout and Beamrider.