Sample-Efficient Deep RL with Generative Adversarial Tree Search

Kamyar Azizzadenesheli [1] **Weitang Liu** [2] **Zachary C. Lipton** [3] **Animashree Anandkumar** [4]

Abstract

We propose Generative Adversarial Tree Search (GATS), a sample-efficient Deep Reinforcement Learning (DRL) algorithm. While Monte Carlo Tree Search (MCTS) is known to be effective for search and planning in RL but, it is often sample-inefficient and therefore expensive to apply in practice. **In this work**, we train Generative Adversarial Networks (GANs) to model an environment's dynamics and train a predictor to learn the reward function. While typical DRL algorithms estimate the Q function or optimize directly over a parameterized policy, we exploit the collected data through interaction with the environment to train both a reward predictor conditional GAN that simulated state transitions. During planning, we deploy finite depth MCTS, using the trained generative model for the tree search and estimated Q value for the leaves, in order to find the best policy. We theoretically show that GATS improves the bias-variance trade-off in DRL. On the Atari game Pong, GATS significantly reduces the bias in Q estimates and leads to a drastic reduction of sample complexity of DQN by a factor of 200%.

1. Introduction

The earliest and best-publicized applications of deep reinforcement learning (DRL) involve Atari games (Mnih et al., 2015) and the board game of Go (Silver et al., 2016), where collecting samples is cheap due to their simulated environments. In such scenarios, DRL can be combined with Monte-Carlo tree search (MCTS) methods (Kearns et al., 2002; Kocsis & Szepesvári, 2006) for efficient planning. On the emulator, the agent executes roll-outs and finds suitable policies.

[1]University of California, Irvine [2]University of California, Davis [3]CMU [4]Caltech. Correspondence to: Kamyar Azizzadenesheli <kazizzad@uci.edu>, Weitang Liu <wetliu@ucdavis.edu>, Zachary C. Lipton <zlipton@cmu.edu>, Animashree Anandkumar <animakumar@gmail.com>.

Accepted at the FAIM workshop "Prediction and Generative Modeling in Reinforcement Learning", Stockholm, Sweden, 2018. Copyright 2018 by the author(s).

In real-world applications, however, collecting samples takes considerable time and effort, e.g. robotics (Levine et al., 2016) and dialogue systems (Lipton et al., 2016). In such scenarios, the agent typically cannot access either the environment model or its corresponding simulator. Thus, the massive sample complexity intruded in MCTS cannot be carried out in such scenarios. In this work, we propose to utilize the samples to learn the environment model and build a simulator, on which MCTS can be performed. MCTS on the imagined world has been studied in the field of psychology, where it is widely accepted that humans make decisions, in part, by imagining the future and deliberating the decisions (Schacter et al., 2012).

Recently, Generative Adversarial Networks (GANs) (Goodfellow et al., 2014) have emerged as a popular tool for generative modeling, especially with high-dimensional data such as images. Unlike previous approaches to image generation which typically produced blurry images due to optimizing an L1 or L2 objective, the GAN produces crisp images. GANs have since been extended for conditional generation, for instance generating an image conditioned on a label (Mirza & Osindero, 2014) and video predictions (Mathieu et al., 2015). Recently, Isola et al. (2017) employed the GAN objective for image-to-image translation, termed as PIX2PIX, e.g., generating a realistic photograph based on a semantic segmentation map.

In this work, we propose Generative Adversarial Tree Search (GATS), a sample-efficient DRL algorithm which exploit the benefits from both model free and model based worlds in RL. For model free part, we extend the notion of DQN (Mnih et al., 2015) [1] while for model based part, we develop a MCTS based algorithm running on learned model dynamics and reward process.

Generative Dynamics Model (GDM). The states in Arcade Learning Environment (Bellemare et al., 2013), Atari games, consist of frames and thus the transition function lends itself to approximation via a conditional GAN trained to predict the next frame based on the previous frames and actions. For GDM we extend the PIX2PIX architecture so that we can condition on both images and actions.

Reward Predictor (RP). Moreover, we train a (RP) to pre-

[1]One could easily use a pick another DRL algorithm

dicts the expected reward at each state-action pair. For planning, GATS deploys the MCTS method, rolling out over a prespecified depth, and at the leaf it uses the estimated Q-function as a value for the leaf state[2].

We theoretically analyze the sources of estimation error in GATS expected return estimator. We study the bias-variance trade-off and show that the bias in DQN estimator exponentially decays while the variance grows as the depth of MCTS grows. Empirically, we validate GATS on a Pong. We study the bias in the Q estimate of DQN and experience GATS, with even one step lookahead (depth one) significantly helps to remove the unpleasant effect of the bias. This leads to a drastic reduction of sample complexity of DQN by a factor of 2. We find that going to higher depths does not appear to confer significantly more benefits on Pong.

2. Related Work

The exploration-exploitation trade-off is extensively studied in RL literature (Kearns & Singh, 2002; Brafman & Tennenholtz, 2003; Asmuth et al., 2009). Jaksch et al. (2010) investigates the regret analysis of MDPs where the Optimism in Face of Uncertainty (OFU) principle is applied to guarantee a high probability regret upper bound. Azizzadenesheli et al. (2016a) deploys OFU in order to propose a high probability regret upper bound for Partially Observable MDPs where spectral methods (Anandkumar et al., 2014) is deployed for learning the environment model. Furthermore, Bartók et al. (2014) tackles the general case of partial monitoring games and provides a minimax regret guarantee that is polynomial in certain dimensions of the problem.

While theoretical RL addresses the trade-off in exploration-exploitation, this problem is still prominent in empirical reinforcement learning research (Mnih et al., 2015; Abel et al., 2016; Azizzadenesheli et al., 2016b). On the empirical side, the recent success in the video games has sparked a flurry of research interest. For example (Cuayáhuitl, 2016; Fatemi et al., 2016; Wen et al., 2016) investigate DRL for dialogue policy learning, with (Lipton et al., 2018) addressing the efficiency of exploration. To combat the sample complexity shortcoming, designing an efficient exploration strategy in DRL has emerged as an active research topic, e.g. optimism Ostrovski et al. (2017) and Thompson Sampling (Osband et al., 2016; Lipton et al., 2018; Azizzadenesheli et al., 2018). Minimizing the Bellman residual using Bootstraps of Q-function has been the core of value based DRP methods (Mnih et al., 2015; Van Hasselt et al., 2016). It has been extensively studied that minimizing the Bellman residual provides a biased estimator of the value function (Antos et al., 2008). In order to mitigate this bias, DQN propose to update the target value less frequent than the rest of the model. This tweak reduces the bias in the value estimator but significantly increases the sample complexity. On the other hand, Monte Carlo sampling strategies, (Kearns et al., 2002; Kocsis & Szepesvári, 2006) have been proposed as an efficient methods for planning, but still suffers from high sample complexity in real world application.

Recently, there have been studies of video prediction in video games (Oh et al., 2015) where the future trajectory of farms are predicted where the quality of the generated frames are measured by training DQN on them. This proposed video prediction model is deployed in (Weber et al., 2017) where an encoder model is provided to encode the generated trajectories into an abstract representation which is used as an additional input to the policy model. They validate their methods on a small puzzle world, Sokoban, and, further, show the capability of their model on multi-task learning in miniPacman environment. Further work on state prediction studies learning a transition model on encoded state representation (Oh et al., 2017) where this approach does not produce much beyond modest gains on Atari games. A similar approach also has been studied in robotics (Wahlström et al., 2015).

3. Preliminaries

An infinite horizon γ-discounted MDP M is a tuple $\langle \mathcal{X}, \mathcal{A}, T, R, P_0, \gamma \rangle$, with state space $\mathcal{X}$, action space $\mathcal{A}$, and the transition kernel $T : x, a \rightarrow \Delta_x$, accompanied with $[0,1]$-bounded reward function of $R : x, a \rightarrow \Delta_r$, where $0 \leq \gamma < 1$. The agent objective is to optimize the overall expected discounted reward over its policy $\pi := \mathcal{X} \rightarrow \mathcal{A}$ i.e. $\eta^* := \eta(\pi^*) = \max_\pi \lim_{N\rightarrow\infty} \mathbb{E}_\pi \left[\sum_{t=0}^{N} \gamma^t r_t | x_0 \sim P_0\right]$. Let $Q_\pi(x,a)$ denote the average discounted reward under policy π starting from state-action x, a.

$$Q_\pi(x,a) := \lim_{N\rightarrow\infty} \mathbb{E}_\pi \left[\sum_{t=0}^{N} \gamma^t r_t | x_0 = x, a_0 = a\right]$$

Since, in RL, the environment is (partially) unknown for the agent, the agent needs to learn the Q. The property of minimizing the Bellman residual as a method to lean Q

$$\mathcal{L}(Q) = \mathbb{E}_\pi \left[\left(Q(x,a) - r - \gamma Q(x',a')\right)^2\right] \qquad (1)$$

has been studied in (Lagoudakis & Parr, 2003; Antos et al., 2008) where, the tuple (x, a, r, x') consists of consecutive samples under behavioral policy π. For a given pair of state and action (x, a), we aim to minimize

$$\left(Q(x,a) - \mathbb{E}_\pi \left[r + \gamma Q(x',a') | \, x, a\right]\right)^2 \qquad (2)$$

in order to minimizing the objective of Eq. 2, a double sampling is required to estimate the inner expectation. To

[2]Potentially one can adapt GATS by plugging the GDM into a preferred planning method, e.g. UCT, MCTS, Policy Gradient

avoid the cost of the double sampling, a common approach is to, instead, minimize the Bellman residual, Eq. 1

$$\mathbb{E}_\pi\left[\left(Q(x,a)-(r+\gamma Q(x',a'))\right)^2\Big|x,a\right]$$
$$=\left(Q(x,a)-\mathbb{E}_\pi\left[r+\gamma Q(x',a')\Big|x,a\right]\right)^2$$
$$+\operatorname{Var}_\pi\left(r+\gamma Q(x',a')\Big|x,a\right) \tag{3}$$

Where minimizing the Bellman residual is equivalent to minimizing Eq. 2 and a addition term as the variance which is not desirable. In order to leverage this bias, Mnih et al. (2015) deploy the notion of target value and network, then minimizes:

$$\mathcal{L}(Q,Q^{target})=\mathbb{E}_\pi\left[\left(Q(x,a)-r-\gamma Q^{target}(x',\hat{a})\right)^2\right] \tag{4}$$

In addition to this bias, a further statistical bias due to limited capacity of network, optimization algorithm, and model mismatch need to be considered.

4. Generative Adversarial Tree Search

We propose Generative Adversarial Tree Search (GATS) as a (more) sample-efficient DRL algorithm. GATS builds upon DQN by reusing the experiences in the replay buffer to learn a reward model RP, and model dynamics GDM. Afterward, GATS deploys Monte Carlo tree search for planing.

Learning the model dynamics and the reward predictor. In order to learn the model dynamics, we propose GDM, parametrized by θ^{GDM}, as an extension to the architecture of PIX2PIX, an image to image model (Isola et al., 2017). The input to GDM is a pair of state (four consecutive frames) and a sequence of action actions where GDM generates the successor frames. We train GDM by sampling mini-batches of experiences from the DQN replay buffer. Simultaneously, we train RP, parameterized with θ^{RP}, using samples in the replay buffer. For Atari games, rewards are $\{-1,0,1\}$.

Planning For planing, GATS deploys Monte Carlo tree search with a given depth H using GDM and RP, instead of Atari emulator, then uses a Q-function to estimate the maximum expected return at the leaf nodes Fig. 5.

Bias and Variance trade-off. In the previous sections, we studied the objective function used in DQN, Eq. 4, which is inherently a biased estimator (Antos et al., 2008). In the latter section we show how big these biases can be in practice. In addition to DQN and statistical biases, the learned Q might suffer from variance due to low samples regime in the sequential regressions defined in DQN. Therefore, let e_Q denote the upper bound on estimation error in Q function

$$\left|Q(x,a)-\tilde{Q}(x,a)\right|\leq e_Q\ ,\ \forall x,a \tag{5}$$

where $\tilde{Q}(x,a)=\mathbb{E}\left[r+\max_{a'}Q(x',a')\right]$. For a given roll-out policy π_r, using GDM, RP, and estimated Q, the expected return $\xi_p(\pi_r,x)$, starting from state x, $\xi_p(\pi_r,x)$ (the subscript p stands for prediction)

$$\mathbb{E}_{\pi_r,\text{GDM},\text{RP}}\left[\sum_{h=0}^{H-1}\gamma^h\hat{r}_h+\gamma^H\max_a\hat{Q}(\hat{x}_H,a)\Big|x\right] \tag{6}$$

Since this expectation is not under the real environment, given GDM,RP and Q estimate, GATS efficiently estimates this expected return without any interaction with the real environment. Let $\xi(\pi_r,x)$ denote the same quantity under the ground truth model

$$\xi(\pi_r,x):=\mathbb{E}_{\pi_r}\left[\sum_{h=0}^{H-1}\gamma^h r_h+\gamma^H\max_a\tilde{Q}(\hat{x}_H,a)\Big|x\right]$$

If $\forall x,x',\hat{x},\hat{x}',a\in\mathcal{X},\mathcal{A}$ the RP error is upper bounded by $\sum_a\left|\left(r(x,a)-\hat{r}(\hat{x},a)\right)\right|\leq e_R$ and GDM is upper bounded by $\sum_{x'}\left|\left(T(x'|x,a)-\hat{T}(\hat{x}'|x,a)\right)\right|\leq e_T$ where $\hat{T}$ and $\hat{r}$ are the estimated transition kernel and reward function[3] then we have:

Theorem 1. *[Bias-Variance trade-off] If* GATS *is run to estimate the Q function using* DQN *procedure, and learn the model of the environment using* GDM *and* RP*, then the deviation in estimating $\xi_p(\pi_r,x)$ is bounded as, $\forall$ x and π_r*

$$|\xi_p(\pi_r,x)-\xi(\pi_r,x)|$$
$$\leq\gamma^He_Q+\frac{\gamma^H}{1-\gamma}He_T+\frac{1-\gamma^H}{1-\gamma}(e_T+e_R). \tag{7}$$

The Thm. 1 provides an insight into the contribution of each source of error into the GATS prediction $\xi_p(\pi_r,x)$. The exponential vanishing error in Q estimates comes at the cost of variances in the model estimation. Therefore, the agent might be able to choose H, given estimated error, in such a way to minimize the estimation error.

5. Experiments

We study the performance of GATS on a Atari game *Pong* using OpenAI Gym (Brockman et al., 2016). The DQN architecture and the game design choices are fully borrowed from (Mnih et al., 2015). The architecture of GDM is inspired by PIX2PIX network (Isola et al., 2017). We deploy the proposed U-Net model as the generator. It consists of five convolutional layers in order to construct the bottle neck where we concatenate it with the input action and followed with five de-convolutional layers to generate next

[3]In the latter sections, we emphatically study e_R and e_T, Fig. 4 and observe their surprisingly small error in practice, e.g. RP, on average, makes less than two mistakes per entire episode in Pong

frames. The discriminator model has three convolutional layers where we also concatenate the input action with the output of these layers which is followed with a fully connected layer. The RP is a simple model with 3 outputs. It consist of 3 convolutional layers, where, similar to the discriminator, we concatenate the input action with the output of these 3 layers which is also followed with a fully connected layer. [4] We train GDM using mini-batches of size 128 and update it every 128 decision steps of GATS. Each sample of mini-batch consists of a trajectory of length $H+1$ where GDM generates the simulated future frames using its self generated frames. We update RP every 16 decision steps.

Bias-Variance of Q_θ. To observe the existing bias and variance in Q_θ, we run solely DQN on the game Pong, for $20M$ time steps. Fig. 1 express 4 consecutive frames where the agent receive a negative score and Table. 1 shows the estimated Q values by DQN for these steps. As we observe in Fig. 1 and Table. 1, at the time step t, the estimated Q value of all the actions are almost the same. The agent makes *down* decision and the environment goes to the next state. The second row of Table. 1 expresses the Q value of the actions at this new state. Since this transition does not carry any reward and discount factor is close to 1, ($\gamma = 0.99$) we expect the max Q values at time step $t+1$ to be close the Q values of action *down* but it is very different. On the other hand it is acceptable that at time step $t+2$ the expected Q values are small, but since the agent receives its negative score at times step $t+3$ we do not expect that the estimated Q at time step $t+3$ to be larger than time step $t+2$.

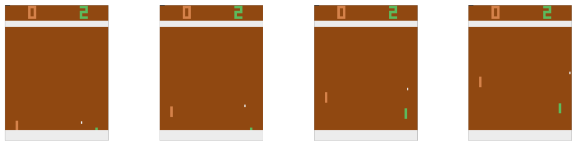

Figure 1. The sequence of four consecutive decision states, $t, t+1, t+2, t+3$ from left to right, where the agent losses a point.

Table 1. The Q function, learned by DQN, when the agent scores a negative point

	Action space		
Steps	stay	up	down
t	4.3918	4.3220	4.3933
$t+1$	2.7985	2.8371	2.7921
$t+2$	2.8089	2.8382	2.8137
$t+3$	3.8725	3.8795	3.8690

We run GATSs with $1, 2, 3,$ and 4 steps lookahead $GATS1, GATS2, GATS3, GATS4$ for $20M$ time steps and show its improvement over DQN Fig. 2. Fig. 2right shows the RP prediction accuracy. We observe when the transition phase at decision step $1M$ occurs Fig. 2, the RP model miss-classifies the positive rewards. But it rapidly adapts to this shift and reduces the classification error to less than 2 errors per episode.

As DRL methods are data hungry, we can re-use the data to efficiently learn the model dynamics. Fig. 4 shows how accurate the GDM can generate next 9 frames just conditioning on the first frame and the trajectory of actions. This trajectory is generated at decision step $150k$.

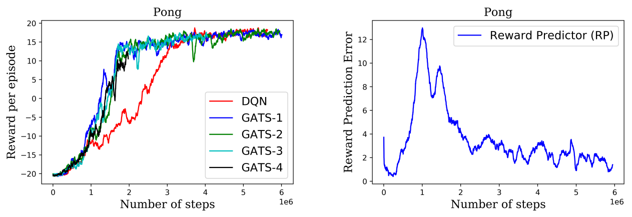

Figure 2. left:GATS learns a better policy faster than plain DQN (2 times faster). GATSk denotes GATS of depth k. *right*: Accuracy of RP. Interestingly, at 1M, when the policy start to change, the acc of RP increases

6. Conclusion

In this work, we proposed GATS a sample efficient DRL paradigm which bridge the gap between model free and model based (black box model). GATS learns the model dynamics while learning the Q-function and used the learned model, as a black box which it queries form, to roll-out different policies in order to find a best policy.

One of the significance of the GATS model is its flexibility. GATS consists of a few blocks: (i) Value learning; we deployed DQN, (ii) Planning; we used pure Monte Carlo sampling, (iii) reward predictor,RP; we used a simple 3-class classifier; (iv) model dynamics, GDM: we extend the PIX2PIX architecture and use it to generate frames.

Practically, one can easily deploy any other methods for each of this blocks. For instance, for i, one can use DDQN (Van Hasselt et al., 2016). For the planning, ii, upper confidence bound tree search (UTC)(Kocsis & Szepesvári, 2006), or policy gradient methods (Kakade, 2002; Schulman et al., 2015). Moreover, if the reward models has a continues distribution, we are interested in mean reward, then any regression method can be useful for reward learning. Last, but not the least, in order to learn model dynamics, one can choose any other image generating models.

Furthermore, this work can be extended to λ-return setting where mix of n steps are acquired. These mentioned feasibilities in GATS are helpful to easily adapt it to the different domains and problems.

[4]We are planning to release the code when we finalize the results for the rest of Atari Games

Acknowledgments

K. Azizzadenesheli is supported in part by NSF Career Award CCF-1254106 and Air Force FA9550-15-1-0221. A. Anandkumar is supported in part by Microsoft Faculty Fellowship, Google Faculty Research Award, Adobe Grant, NSF Career Award CCF-1254106, and AFOSR YIP FA9550-15-1-0221.

References

Abel, David, Agarwal, Alekh, Diaz, Fernando, Krishnamurthy, Akshay, and Schapire, Robert E. Exploratory gradient boosting for reinforcement learning in complex domains. *arXiv*, 2016.

Anandkumar, Animashree, Ge, Rong, Hsu, Daniel, Kakade, Sham M, and Telgarsky, Matus. Tensor decompositions for learning latent variable models. *The Journal of Machine Learning Research*, 15(1):2773–2832, 2014.

Antos, András, Szepesvári, Csaba, and Munos, Rémi. Learning near-optimal policies with bellman-residual minimization based fitted policy iteration and a single sample path. *Machine Learning*, 2008.

Asmuth, John, Li, Lihong, Littman, Michael L, Nouri, Ali, and Wingate, David. A bayesian sampling approach to exploration in reinforcement learning. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, 2009.

Azizzadenesheli, Kamyar, Lazaric, Alessandro, and Anandkumar, Animashree. Reinforcement learning of pomdps using spectral methods. In *Proceedings of the 29th Annual Conference on Learning Theory (COLT)*, 2016a.

Azizzadenesheli, Kamyar, Lazaric, Alessandro, and Anandkumar, Animashree. Reinforcement learning in rich-observation mdps using spectral methods. *arXiv preprint arXiv:1611.03907*, 2016b.

Azizzadenesheli, Kamyar, Brunskill, Emma, and Anandkumar, Animashree. Efficient exploration through bayesian deep q-networks. *arXiv preprint arXiv:1802.04412*, 2018.

Bartók, Gábor, Foster, Dean P, Pál, Dávid, Rakhlin, Alexander, and Szepesvári, Csaba. Partial monitoringclassification, regret bounds, and algorithms. *Mathematics of Operations Research*, 2014.

Bellemare, Marc G, Naddaf, Yavar, Veness, Joel, and Bowling, Michael. The arcade learning environment: An evaluation platform for general agents. *J. Artif. Intell. Res.(JAIR)*, 2013.

Brafman, Ronen I and Tennenholtz, Moshe. R-max-a general polynomial time algorithm for near-optimal reinforcement learning. *The Journal of Machine Learning Research*, 3:213–231, 2003.

Brockman, Greg, Cheung, Vicki, Pettersson, Ludwig, Schneider, Jonas, Schulman, John, Tang, Jie, and Zaremba, Wojciech. Openai gym, 2016.

Cuayáhuitl, Heriberto. Simpleds: A simple deep reinforcement learning dialogue system. *arXiv:1601.04574*, 2016.

Fatemi, Mehdi, Asri, Layla El, Schulz, Hannes, He, Jing, and Suleman, Kaheer. Policy networks with two-stage training for dialogue systems. *arXiv:1606.03152*, 2016.

Goodfellow, Ian, Pouget-Abadie, Jean, Mirza, Mehdi, Xu, Bing, Warde-Farley, David, Ozair, Sherjil, Courville, Aaron, and Bengio, Yoshua. Generative adversarial nets. In *Advances in neural information processing systems*, pp. 2672–2680, 2014.

Isola, Phillip, Zhu, Jun-Yan, Zhou, Tinghui, and Efros, Alexei A. Image-to-image translation with conditional adversarial networks. *arXiv preprint*, 2017.

Jaksch, Thomas, Ortner, Ronald, and Auer, Peter. Near-optimal regret bounds for reinforcement learning. *Journal of Machine Learning Research*, 2010.

Kakade, Sham M. A natural policy gradient. In *Advances in neural information processing systems*, 2002.

Kearns, Michael and Singh, Satinder. Near-optimal reinforcement learning in polynomial time. *Machine Learning*, 49(2-3):209–232, 2002.

Kearns, Michael, Mansour, Yishay, and Ng, Andrew Y. A sparse sampling algorithm for near-optimal planning in large markov decision processes. *Machine Learning*, 49 (2-3):193–208, 2002.

Kocsis, Levente and Szepesvári, Csaba. Bandit based monte-carlo planning. In *Machine Learning: ECML 2006*, pp. 282–293. Springer, 2006.

Lagoudakis, Michail G and Parr, Ronald. Least-squares policy iteration. *Journal of machine learning research*, 4 (Dec):1107–1149, 2003.

Levine et al., Sergey. End-to-end training of deep visuomotor policies. *JMLR*, 2016.

Lipton, Zachary C, Gao, Jianfeng, Li, Lihong, Chen, Jianshu, and Deng, Li. Combating reinforcement learning's sisyphean curse with intrinsic fear. *arXiv preprint arXiv:1611.01211*, 2016.

Lipton, Zachary C, Gao, Jianfeng, Li, Lihong, Li, Xiujun, Ahmed, Faisal, and Deng, Li. Efficient exploration for dialogue policy learning with bbq networks & replay buffer spiking. *AAAI*, 2018.

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

Mirza, Mehdi and Osindero, Simon. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.

Mnih, Volodymyr, Kavukcuoglu, Koray, Silver, David, Rusu, Andrei A, Veness, Joel, Bellemare, Marc G, Graves, Alex, Riedmiller, Martin, Fidjeland, Andreas K, Ostrovski, Georg, et al. Human-level control through deep reinforcement learning. *Nature*, 2015.

Oh, Junhyuk, Guo, Xiaoxiao, Lee, Honglak, Lewis, Richard L, and Singh, Satinder. Action-conditional video prediction using deep networks in atari games. In *Advances in Neural Information Processing Systems*, 2015.

Oh, Junhyuk, Singh, Satinder, and Lee, Honglak. Value prediction network. In *Advances in Neural Information Processing Systems*, pp. 6120–6130, 2017.

Osband, Ian, Blundell, Charles, Pritzel, Alexander, and Van Roy, Benjamin. Deep exploration via bootstrapped dqn. In *Advances in Neural Information Processing Systems*, 2016.

Ostrovski, Georg, Bellemare, Marc G, Oord, Aaron van den, and Munos, Rémi. Count-based exploration with neural density models. *arXiv preprint arXiv:1703.01310*, 2017.

Schacter, Daniel L, Addis, Donna Rose, Hassabis, Demis, Martin, Victoria C, Spreng, R Nathan, and Szpunar, Karl K. The future of memory: remembering, imagining, and the brain. *Neuron*, 76(4):677–694, 2012.

Schulman, John, Levine, Sergey, Abbeel, Pieter, Jordan, Michael, and Moritz, Philipp. Trust region policy optimization. In *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, 2015.

Silver et al., David. Mastering the game of go with deep neural networks and tree search. *Nature*, 2016.

Van Hasselt, Hado, Guez, Arthur, and Silver, David. Deep reinforcement learning with double q-learning. In *AAAI*, 2016.

Wahlström, Niklas, Schön, Thomas B, and Deisenroth, Marc Peter. From pixels to torques: Policy learning with deep dynamical models. *arXiv preprint arXiv:1502.02251*, 2015.

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

Wen, Tsung-Hsien, Gasic, Milica, Mrksic, Nikola, Rojas-Barahona, Lina M, Su, Pei-Hao, Ultes, Stefan, Vandyke, David, and Young, Steve. A network-based end-to-end trainable task-oriented dialogue system. *arXiv:1604.04562*, 2016.

A. Appendix

Bias-Variance of Q_θ. In Fig. 3 and Table. 2 we investigate the case that the agent catches the ball. The ball is going to the right and agent needs to catch it. At times step t, the paddle is not on the direction of ball velocity and as it is shown in Table. 2, the optimal action is going *down*. But a closer look at the estimated Q value of action *up* reveals that the Q value for both of this actions are unreasonably close and action *up* might end up to losing the point.

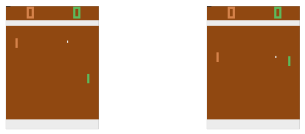

Figure 3. The sequence of two consecutive decision states, $t-1, t$ from left to right, where the agent catch the ball.

Table 2. The Q function, learned by DQN, when the agent catches the ball

	Action space		
Steps	stay	up	down
t	1.5546	4.5181	4.5214

Lastly we studied the existing errors in the estimation of the Q function using DQN. As, for example, it is expressed in Table.1, if the agent could roll-out before making decision, even-if it is just one step look ahead) it could realize the negative return of making action down. The positive effect of the roll-out is more significant in earlier stage of learning Q where the estimation of Q is more off.

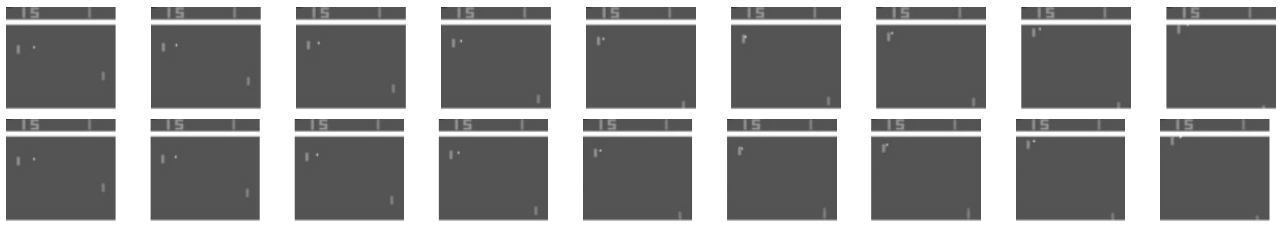

Figure 4. First row: A sequence of real frames. Second row: a corresponding sequence of generated frames

B. Proof of Thm. 1

We decompose the error in estimation $\xi_p(\pi_r, x)$ into two parts. The first part, also caused by the first term in the right hand side of Eq. 6, carries an error in modeling of the environment and depends on the deficiency in the perfectness of RP and GDM models. The second part, also caused by the second term in the right hand side of Eq. 6, is mostly due to the bias and variance in DQN estimate of Q-function e_Q but still depends on the GDM and the RP since the the distribution of $\hat{x}_H$ is depends on GDM. Therefore, for the second term, by adding and subtraction this term, $\mathbb{E}_{\pi_r}\left[\gamma^H \max_a \hat{Q}(\hat{x}_H, a)\right]$, we have;

$$\left|\mathbb{E}_{\pi_r,\text{GDM},\text{RP}}\left[\gamma^H \max_a \hat{Q}(\hat{x}_H, a)\Big|x\right] - \mathbb{E}_{\pi_r}\left[\gamma^H \max_a Q(x_H, a)\Big|x\right]\right| \leq \gamma^H e_Q + \frac{\gamma^H}{1-\gamma}\sum_{x_H}\left|P(x_H|x,\pi_r) - \hat{P}(\hat{x}_H|x,\pi_r)\right| \tag{8}$$

The term $\frac{1}{1-\gamma}$ appears due to the fact that the maximum possible Q is not grater that $\frac{1}{1-\gamma}$. In order to bound $\left|P(x_H|x,\pi_r) - \hat{P}(\hat{x}_H|x,\pi_r)\right|$, we need to expand them even further to their pieces. For instance, for $P(x_H|x,\pi_r)$, we have

$$P(x_H|x,\pi_r) := \sum_{x_i, a_i, \forall i\in[1,.,H-1]} T(x_1|x,a_1)\pi_r(a_1|x)\prod_{i=2}^{H-1} T(x_i|x_{i-1},a_i)\pi_r(a_i|x_{i-1})T(x_H|x_{H-1},a_H)\pi_r(a_H|x_{H-1}) \tag{9}$$

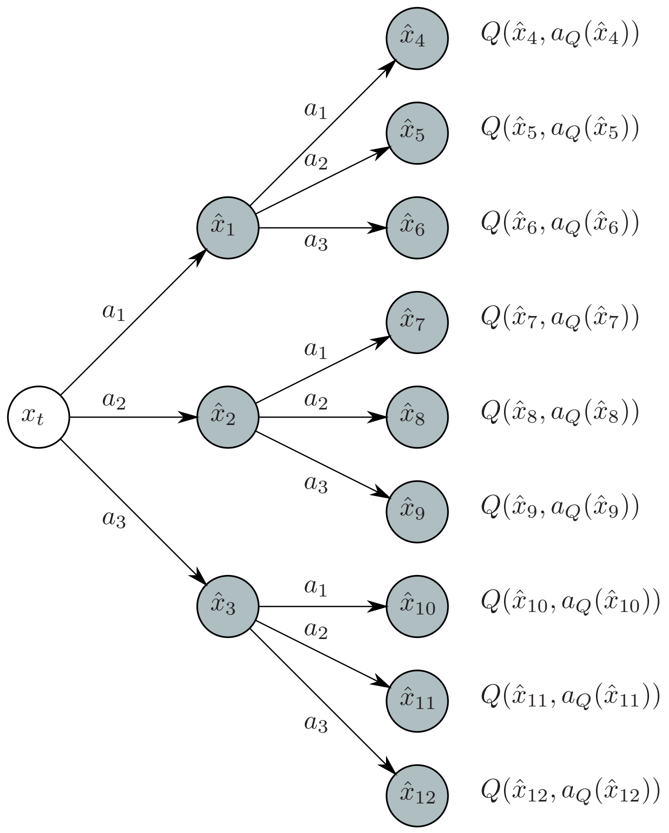
x_t
a_1
a_2
a_3
$\hat{x}_1$
$\hat{x}_2$
$\hat{x}_3$
$\hat{x}_4$
$\hat{x}_5$
$\hat{x}_6$
$\hat{x}_7$
$\hat{x}_8$
$\hat{x}_9$
$\hat{x}_{10}$
$\hat{x}_{11}$
$\hat{x}_{12}$
$Q(\hat{x}_4, a_Q(\hat{x}_4))$
$Q(\hat{x}_5, a_Q(\hat{x}_5))$
$Q(\hat{x}_6, a_Q(\hat{x}_6))$
$Q(\hat{x}_7, a_Q(\hat{x}_7))$
$Q(\hat{x}_8, a_Q(\hat{x}_8))$
$Q(\hat{x}_9, a_Q(\hat{x}_9))$
$Q(\hat{x}_{10}, a_Q(\hat{x}_{10}))$
$Q(\hat{x}_{11}, a_Q(\hat{x}_{11}))$
$Q(\hat{x}_{12}, a_Q(\hat{x}_{12}))$

Figure 5. Roll-out of depth two starting from state x_t. Here $\hat{x}$'s are the generated states by GDM. $Q(x, a(x))$ denotes the predicted value of state x choosing the greedy action $a_Q(x) := argmax_{a' \in \mathcal{A}} Q(x, a')$.

Again, with adding and subtracting trick, the difference can be written as follows;

$$\sum_{x_H}\Big|P(x_H|x,\pi_r) - \hat{P}(\hat{x}_H|x,\pi_r)\Big| = \sum_{x_i,a_i,\forall i\in[H]} \Big|T(x_1|x,a_1) - \hat{T}(\hat{x}_1|x,a_1)\Big|\,\pi_r(a_1|x)\prod_{i=2}^{H} T(x_i|x_{i-1},a_i)\pi_r(a_i|x_{i-1})$$
$$+\sum_{j=2}^{H}\sum_{x_h,a_h,\forall i\in[H]}\Big(\hat{T}(\hat{x}_1|x,a_1)\Big)\,\pi_r(a_1|x)\Big|T(x_j|x_{j-1},a_j) - \hat{T}(\hat{x}_j|x_{j-1},a_j)\Big|$$
$$\prod_{h=2}^{j-1} T'(\hat{x}_h|x_{h-1},a_h)\pi_r(a_h|\hat{x}_{i-1})\prod_{h=j+1}^{H} T(x_h|x_{h-1},a_h)\pi_r(a_h|x_{h-1}) \tag{10}$$

Since for any pair of state and action (x, a) we have $\sum_{x'}\Big|\Big(T(x'|x,a) - T'(\hat{x}'|x,a)\Big)\Big| \leq e_T$, substituting e_T in Eq. 10, and then marginalizing the joint distributions we get

$$\sum_{x_H}\Big|P(x_H|x,\pi_r) - \hat{P}(\hat{x}_H|x,\pi_r)\Big| \leq He_T \tag{11}$$

Now, we can interpret that GATS significantly (exponentially in depth) reduces the bias and variance, e_Q, effect of the estimated Q by DQN, $\gamma^H e_Q$. At the same time, since the maximum possible Q is less than or equal to $\frac{1}{1-\gamma}$, the error in the second term of Eq. 8 caused by GDM goes down as $\frac{\gamma^H}{1-\gamma}He_T$.

As it is mentioned before, another source of error in estimating $\xi_p(\pi_r, x)$ comes from the first term in the right hand side of

Eq. 6, which is due to deficiency in the RP and GDM models.

$$\left|\mathbb{E}_{\pi_r,\text{GDM},\text{RP}}\left[\sum_{h=0}^{H-1}\gamma^h\hat{r}_h\right]-\mathbb{E}_{\pi_r}\left[\sum_{h=0}^{H-1}\gamma^h r_h\right]\right| \tag{12}$$

In order to bound this quantity, we use the same decomposition procedure used in Eq. 10 and;

$$\left|\mathbb{E}_{\pi_r,\text{GDM},\text{RP}}\left[\sum_{h=0}^{H-1}\gamma^h\hat{r}_h\right]-\mathbb{E}_{\pi_r}\left[\sum_{h=0}^{H-1}\gamma^h r_h\right]\right| \leq \sum_{i}^{H-1}\gamma^i e_T+\sum_{i}^{H-1}\gamma^i e_R=\frac{1-\gamma^H}{1-\gamma}(e_T+e_R) \tag{13}$$

The theorem follows up.

C. GATS Algorithm

Ideally, if the GDM model is perfect at generating frames (the dynamics can be distorted), i.e. the space generated frames is same as the real frames, for the leaf nodes x_H, we could use $\max_a Q(x_H,a;\theta)$, learned by the DQN model on real frames, in order to assign a value to the leaf nodes. But in practice, instead of x_H, we have access to $\hat{x}_H$, a generated state that is perceptually is similar to x_H (Fig. 4), but from the perspective of Q_θ, they might not be similar since, during the course of training of Q_θ, there is no generated data from GDM in the training set. Therefore we train another Q-network, parameterized with θ', in order to provide the similar Q-value as Q_θ for perceptually similar states. To train $Q_{\theta'}$, we utilize the same procedure used in DQN. We trained $Q_{\theta'}$ in order to match the target value, provided by $Q_{\theta^{target}}$. As described in Fig. 6, for the state x_{t+H} and its corresponding generated state $\hat{x}_H$, we update $Q(\hat{x}_H,a_{t+H};\theta')$ using a similar loss to that used in DQN updates;

$$\left(Q(\hat{x}_H,a_{t+H};\theta')-r_{t+H}-\gamma\max_{a'}Q(x_{t+H+1},a';\theta^{target})\right)^2$$

and update θ' in order to minimize this loss.

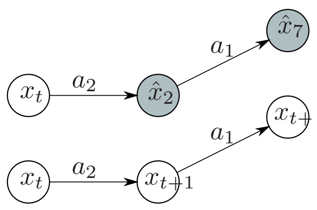

Figure 6. Training GAN and $Q_{\theta'}$ using the longer trajectory of experiences

Algorithm 1 GATS (H)

```
1:  Initialize parameter sets $\theta$, $\theta'$, $\theta^{target}$, $\theta^{\mathrm{GDM}}$, $\theta^{\mathrm{RP}}$
2:  Initialize replay buffer and set counter $=\ 0$
3:  for episode = 1 to $\inf$ do
4:      for $t =$ to the end of episode do
5:          $a_t = roll - out(x_t, H, theta', \theta^{\mathrm{GDM}}, \theta^{\mathrm{RP}})$
6:          Store transition $(x_t, a_t, r_t, x_{t+1})$ in replay buffer
7:          Sample a random minibatch of transitions ($x_\tau$ , $a_\tau$, $r_\tau$, $x_{\tau+1}$) from replay buffer
8:          $y_\tau \leftarrow \left\{ \begin{array}{l} \text{for terminal } s_{\tau+1} : \\ \quad r_\tau \\ \text{for non-terminal } x_{\tau+1} : \\ \quad r_\tau + \max_{a'} Q(x_{\tau+1}, a'; \theta^{target}) \end{array} \right\}$
9:          $\theta \leftarrow \theta - \eta \cdot \nabla_\theta (y_\tau - Q(x_\tau, a_\tau; \theta))^2$
10:         $\theta' \leftarrow \theta' - \eta \cdot \nabla_{\theta'} (y_\tau - Q(\hat{x}_\tau, a_\tau; \theta'))^2$
11:         Update GDM, and RP
12:     end for
13: end for
```