Planning in Dynamic Environments with Conditional Autoregressive Models

Johanna Hansen* [1] **Kyle Kastner*** [2] **Aaron Courville** [2 3] **Gregory Dudek** [1]

Abstract

We demonstrate the use of conditional autoregressive generative models (van den Oord et al., 2016a) over a discrete latent space (van den Oord et al., 2017b) for forward planning with MCTS. In order to test this method we introduce a new environment featuring varying difficulty levels, along with moving goals and obstacles. The combination of high-quality frame generation and classical planning approaches nearly matches true environment performance for our task, demonstrating the usefulness of this method for model-based planning in dynamic environments.

1. Introduction

Planning agents find actions at each decision point by considering future scenarios from their current state against a model of their world (Lavalle, 1998; Kocsis & Szepesvári, 2006; Stentz, 1995; van den Berg et al., 2006). Though typically slower at decision-time than model-free agents, agents which use planning can be configured and tuned with explicit constraints. Planning based methods can also reduce the compounding of errors for sequential decisions by directly testing long term consequences from action choices, balancing exploitation and exploration, and generally limiting issues with long-term credit assignment.

Model-free reinforcement learning approaches are often sample inefficient, requiring millions of steps to jointly learn environment features and a control policy. Agents which employ decision-time planning techniques, on the other hand, do not explicitly require any training prior to decision time. However, to perform well, planning-based agents need a very accurate future model of their environment for evaluating actions. A perfect model of the future to perform forward planning is usually not possible outside of computer games or simulations. In this paper, we demonstrate how we can leverage recent improvements in generative modeling to create powerful dynamics models that can be used for forward planning.

In this paper we discuss an approach for learning conditional models of an environment in an unsupervised manner, and demonstrate the utility of this model for use with decision-time planning in a dynamic environment. Autoregressive models have shown great results in generating raw images, video, and audio (van den Oord et al., 2016a;b; Kalchbrenner et al., 2016), but have generally been considered too slow for use in decision making agents (Buesing et al., 2018). However, in (van den Oord et al., 2017b), the authors show that these autoregressive models can be used as a generative prior over the latent space of discrete encoder/decoder models. Operating over these concise latent representations of the data instead of pixel-space greatly reduces the time needed for generation, making these models feasible for use in decision-making agents.

2. Background

Learning accurate models of the environment has long been a goal in model-based reinforcement learning and unsupervised learning. Recent work has shown the power of learning action-conditional models for training decision-making agents with perceptual models (Ha & Schmidhuber, 2018; Schmidhuber, 2015; Buesing et al., 2018; Oh et al., 2015; Graves, 2013) and combining planning and with environment models (Silver et al., 2016b; Zhang et al., 2018; Pascanu et al., 2017; Guez et al., 2018; Anthony et al., 2017; Guez et al., 2018).

For real-world agents, semantic information is often more relevant than perceptual input for task performance and planning (Luc et al., 2017). Our experimentation over semantic space shows that for our task, a VQ-VAE model greatly outperforms VAE (Kingma & Welling, 2013) reconstructions. Instead of assuming normally distributed priors and posteriors as in a typical VAE architecture, VQ-VAEs learns categorical distributions in the latent space where the samples from the distributions are indexes to an embedding table. Van den Oord et al. (van den Oord et al., 2017b) demonstrates the benefits of learning action-condition and

*Equal contribution [1]Mobile Robotics Lab, School of Computer Science, McGill University, Montréal, Québec, Canada [2]Montréal Institute for Learning Algorithms (MILA), Université de Montréal, Montréal, Québec, Canada [3]CIFAR Fellow. Correspondence to: Johanna Hansen <johanna.hansen@mail.mcgill.ca>.

Accepted at the FAIM workshop "Prediction and Generative Modeling in Reinforcement Learning", Stockholm, Sweden, 2018.

action-independent forward predictions over VQ-VAE latent space. We build upon this work by combining it with a classical method for planning in order to navigate in an environment with numerous dynamic obstacles and a moving target.

We test our forward-model with a powerful anytime planning method, Monte-Carlo Tree Search (MCTS) (Kocsis & Szepesvári, 2006). Given an accurate representation of the future and sufficient time to compute, MCTS performs well (Pepels et al., 2014), even when faced with large state or action spaces. MCTS works by *rolling out* many sequences of actions possible future scenarios to acquire an approximate (Monte Carlo) estimate of the value of taking a specific action from a particular state. For a full overview of MCTS and its many variants, please refer to (Browne & Powley, 2012). MCTS has been used in a wide variety of search and planning problems where a model of the world is available for querying (Silver et al., 2016a; Guo et al., 2014a; Bellemare et al., 2012; Lipovetzky et al., 2015; Guo et al., 2014b). The performance of MCTS is critically dependent on having an accurate forward model of the environment, making it an ideal fit for testing our autoregressive conditional generative forward model.

3. Experiments

We consider a fully-observable task in which an agent must navigate to a dynamic goal location without contact with moving obstacles. At each time step t, the agent realizes an observation o_t and must execute an action a_t. In our experiments, the observation is an image constituting the full view of an action-independent, two-dimensional environment. The action space consists of 8 actions, where each action moves a fixed amount in a specific direction, diagonal included. We learn a conditional forward model of this environment as described in Section 3.2 and query at decision time for action selection with MCTS.

Our problem is similar to those faced by autonomous underwater vehicles (AUVs) navigating in a busy harbor while try to avoid traveling underneath passing ships (Arvind et al., 2013). In order to successfully accomplish this tasks, the robot needs reliable dynamics models of the obstacles (ships) and goals in the environment so it can plan effectively against a realistic estimate of future states.

3.1. Environment Description

We introduce a navigation environment (depicted in the first column of Figure 1) which consists of a configurable world with dynamic obstacles and a moving goal. Movement about the environment is continuous, but collision and goal checking is quantized to the nearest pixel. In each episode the 1×1 size agent and 2×2 size goal are initialized to a random location, and the goal is given a random vector direction and a fixed velocity. The agent must then reach the moving goal within a limited number of steps without colliding with an obstacle. At each timestep the agent has the choice of 8 actions. These actions indicate one of 8 equally spaced angles and a constant speed. In these experiments, we test two agents, one at 0.5 pixels per timestep ($1\times$ goal agent) and one agent at 1 pixels per timestep ($2\times$ goal agent). The goal moves about the environment at a fixed random angle and fixed speed of 0.5 pixels per timestep. The goal also reflects off of world boundaries, making good modeling of goal dynamics important to success.

The environment is divided into obstacle *lanes* which span the environment horizontally. At the beginning of each

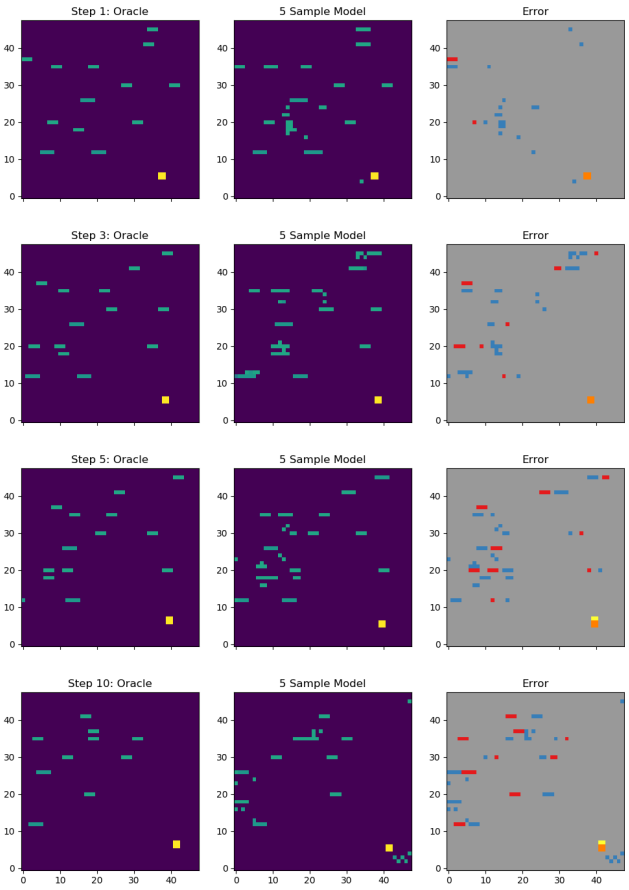

Figure 1. This figure illustrates forward rollout steps by the oracle (left column), our 5 sample model (middle column), and the error in the model (right column). The number of steps from the given state t is indicated in the oracle plot's title. In the first two columns, free space is violet, moving obstacles are cyan, and the goal is yellow. In the third column, we illustrate obstacle error in the model as follows: false negatives (predicted free space where there should be an obstacle) are red and false positives (predicted obstacle where there was free space) are blue. The true goal is plotted in yellow and the predicted goal is plotted in orange (perfect goal prediction is orange).

episode, the lanes are randomly assigned to carry 1 of 5 classes of obstacles and a direction of movement (left to right or right to left). Each obstacle class is parameterized by a color and a distribution which describes average obstacle speed and length. Obstacles maintain a constant speed after entering the environment, pass through the edges of the environment, and are deleted after their entire body exits the observable space. The number of obstacles introduced into the environment at each timestep is controlled by a Poisson distribution, configured by the *level* parameter. For the results reported in this paper we set the level to 6, however there is support for a variety of difficulty settings. At each time step, the observation consists of the agent's current location and the full quantized pixel space including the goal and obstacles.

An agent receives a reward of $+20$ for entering the same pixel-space as the goal and a -20 reward for entering the same pixel-space as an obstacle. Both events cause the episode to end. The agent has a limited number of actions before the game times out, resulting in a reward of 0. This step limit is dependent on the speed of the agent and the size of the grid. For these experiments, the $2\times$ agent has 203 steps and the $1\times$ agent has 407 steps before the game ends.

A key component which makes our approach computationally feasible is that the environments of concern are *not* action conditional, meaning dynamics in the world continue regardless of what actions are chosen. This means that generated future frames can be shared across all rollouts in MCTS, greatly reducing the overall sample cost for the autoregressive model. Combined with the speed improvements from generating in a compressed space given by VQ-VAE, forward generation can be accomplished in reasonable time. It is also possible to take a similar approach in action-conditional spaces, but this would increase the number of needed generations from the model during MCTS rollout by a large amount.

3.2. Model Description

We utilize a two-phase training procedure on the agent-independent, $48 \times 48 \times 1$ environment described in the previous section. First we learn a compact, discrete representation (denoted Z) of individual pixel-space frames with a VQ-VAE model (van den Oord et al., 2017b) with discretized logistic mixture likelihood (Salimans et al., 2017) for the reconstruction loss. In the second stage, an autoregressive generative model, a conditional gated PixelCNN (van den Oord et al., 2016a) is trained to predict one-step ahead Z representations of sequential frames when conditioned on previous Z representations. To introduce Markovian conditions, the conditional gated PixelCNN is fed a spatial conditioning map of 4 past Z encodings, in addition to the current step. The resulting PixelCNN learns a model corresponding to $p(Z_{t_{i,j}}|Z_{t_{<i,<j}}, Z_{t-1}, Z_{t-2}, Z_{t-3}, Z_{t-4})$, where each dimension (i, j) of Z_t is conditioned on all valid dimensions relative to the current position via autoregressive masking, and also conditioned on the previous 4 frames by a spatial conditioning map (van den Oord et al., 2016a) which is fed as input. Combined with the previously trained VQ-VAE decoder this results in a model which generates 1 frame ahead, given 4 previous frames. It is possible to generate an arbitrary number of frames forward given an initial 4 frames, by chaining 1 step generations though we expect results to degrade as forward trajectory lengths increase.

3.3. MCTS Planning

Our MCTS agent is characterized by rollout length, number of rollouts, and temperature. We vary rollout length from 1 to 10, but hold the number of rollouts to 100 and temperature to 0.01 for all experiments. We also use a goal-oriented prior for node selection as described by prior work using PUCT MCTS (Rosin, 2011; Silver et al., 2017). This prior biases tree expansion during rollouts such that actions in the direction of the predicted goal are more likely to be chosen. Adding goal information to the state has been found to improve agents in other scenarios (Sukhbaatar et al., 2017), and we found that this simple prior greatly improved performance compared to a uniform prior, resulting in shorter average rollout lengths.

3.4. Training

The VQ-VAE encoder consists of 4 strided convolutional layers with a kernel size $(4, 4)$ and sizes of 42, 32, 16, 16. The first 3 layers have strides of 2 and the last layer has a stride of 1. This configuration compresses an input size of $48 \times 48 \times 1$ down to a Z space of $6 \times 6 \times 1$. For learning the vector quantization codebook, we set K=512, resulting in a compression of $\frac{48\times48\times3}{6\times6\times9} \approx 21.3$ in bits over each frame, considering there are 6 pixel-values used in the input image (requiring 2^3 bits to encode minimally). The VQ-VAE decoder inverts this process using transpose convolutions, and appropriate stride values which mimic the decoder settings but in reverse order.

Training was performed for 64 epochs with a minibatch size of 32 over $837,270$ example frames which were generated from running the environment. We use an Adam optimizer (Kingma & Ba, 2014) with the learning rate set to $1e-3$, and the discretized mixture of logistics loss (Salimans et al., 2017). From the trained VQ-VAE model, we generate a new dataset consisting of ordered Z values given by our model over 3000 previously unseen episodes which are each 407 frames long. The PixelCNN (van den Oord et al., 2016a) is trained over these generated Zs for 10 epochs with a batch size of 64. We employ categorical cross-entropy loss and the Adam optimizer (learning rate is set to 0.0003) for

Table 1. Performance Comparison over 100 Episodes

ROLLOUT STEPS	1				3				5				10			
TECHNIQUE	G	T	D	S	G	T	D	S	G	T	D	S	G	T	D	S
2×ORACLE	100	0	0	34X±17	100	0	0	36±18	100	0	0	45±28	100	4	0	68±49
2×MID	78	0	22	33±17	88	0	12	40±18	91	2	7	65±40	52	25	23	111±67
2×5 SAMPLES	84	0	16	34±17	**94**	1	5	46±27	89	5	6	75±51	55	23	22	112±70
2×10 SAMPLES	85	0	15	35±18	88	0	12	46±26	89	9	2	76±56	55	31	14	124±68
1×ORACLE	72	25	3	187±151	67	32	1	209±154	60	40	0	224±64	66	34	0	216±156
1×5 SAMPLES	31	3	55	97 ± 107	46	21	33	196±153	41	3	27	259±155	39	46	15	294±143

Table 2. This table compares agents using MCTS for forward planning on varying models (oracle and ours with varying levels of sampling from the generative model), rollout lengths (1, 3, 5 and 10), and agent speed (2X agents are twice as fast as the goal and 1X agents are the same speed as the goal). All agents were tested over the same set of 100 random episodes, with MCTS performing 100 rollouts at each decision time. The values in columns *G*, *T*, and *D* stand for the number of games in which the described agent reached the goal (*G*), ran out of time before reaching the goal (*T*), or died (*D*) by running into an obstacle. The *S* column describes the number of steps completed on average by an agent, calculated only from episodes in which the agent avoided dying (smaller is better), along with the standard deviation. When tested on the same episodes, a random agent reached the goal once at 2X speed and never at 1X speed.

predicting the discrete "label" of each Z dimension. We condition each prediction on a spatial map consisting of the previous 4 frame's Zs (van den Oord et al., 2016a).

4. Performance

Our experiments (see Table 1) demonstrate the feasibility of using conditional autoregressive models for forward planning. Example playout gifs can be found in the code repository at `https:github.com/johannah/trajectories`. We compare agents using our forward model to an agent which has access to an oracle of the environment. The oracle agent is used as an upper-bound on performance, as although this perfect representation of the future environment is not available in realistic tasks it is the theoretical best we can expect generative model to do. In all of the compared models, we first use a mid point "average" estimate from the discretized mixture of logistics distribution, but in those denoted by *sampled*, we also sample an additional 5 or 10 times from the model and take the pixel-wise max of the predicted obstacle values. We find this results in a more conservative, but noisier estimate of the car locations. We take the median location of goal estimates over all of the samples to set the directional MCTS prior.

Errors in the forward predictions (see Figure 1) can cause the agents to make catastrophic decisions, resulting in lower performance when compared to the oracle. False negatives, in particular (shown in red in Figure 1), result in the agent mistaking an obstacle for free space. Some of these mistakes are unavoidable as we step farther from the given state as we can only model obstacles that are in the scene at the current time step. This characteristic limits the efficacy of the lengths we can model forward in time and is a phenomena also discussed in Luc et al. (Luc et al., 2017).

Perhaps unsurprisingly, our results show that the faster (2×) agent had an easier time reaching the goal before running out of time. Agents which utilize longer rollouts were likely hampered by our decision to hold the number of rollouts constant over all of our experiments. Overall, longer rollouts were more likely to die off in their future states and thus often failed to come up with aggressive paths.

Each future timestep prediction with our VQ-VAE + PixelCNN takes approximately 0.4 seconds on a TitanX-Pascal GPU. An average action decision with our best performing agent (2× 5 Samples with 3 step rollouts) takes approximately 1.7 seconds. Beyond using VQ-VAE to reduce the input space to PixelCNN, no other methods for improving the speed of autoregressive generation were employed. Recent publications in this area (van den Oord et al., 2017a; Kalchbrenner et al., 2018; Ramachandran et al., 2017) show massive improvements in generation speed for autoregressive models and are directly applicable to this work.

5. Conclusion

We show that the two-stage pipeline of VQ-VAE (van den Oord et al., 2017b) combined with a PixelCNN prior conditioned on previous frames captures important semantic structure in a dynamic, goal oriented environment. The resulting samples are usable for model-based planning with MCTS over generated future states. Our agent avoids moving obstacles and reliably intercepts a non-stationary goal in the dynamic test environment introduced in this work, demonstrating the efficacy of this approach for planning in dynamic environments.

References

Anthony, Thomas, Tian, Zheng, and Barber, David. Thinking fast and slow with deep learning and tree search. *CoRR*, abs/1705.08439, 2017.

Arvind, Pereira, Jonathan, Binney, Geoffrey, Hollinger, and Gaurav, Sukhatme. Riskaware path planning for autonomous underwater vehicles using predictive ocean models. *Journal of Field Robotics*, 30(5):741–762, 2013. doi: 10.1002/rob.21472.

Bellemare, Marc G., Naddaf, Yavar, Veness, Joel, and Bowling, Michael. The arcade learning environment: An evaluation platform for general agents. *CoRR*, abs/1207.4708, 2012.

Browne, Cb and Powley, Edward. A survey of monte carlo tree search methods. *Intelligence and AI*, 4(1):1–49, 2012. ISSN 1943-068X. doi: 10.1109/TCIAIG.2012.2186810.

Buesing, Lars, Weber, Theophane, Racanière, Sébastien, Eslami, S. M. Ali, Rezende, Danilo Jimenez, Reichert, David P., Viola, Fabio, Besse, Frederic, Gregor, Karol, Hassabis, Demis, and Wierstra, Daan. Learning and querying fast generative models for reinforcement learning. *CoRR*, abs/1802.03006, 2018.

Graves, Alex. Generating sequences with recurrent neural networks. *CoRR*, abs/1308.0850, 2013.

Guez, Arthur, Weber, Théophane, Antonoglou, Ioannis, Simonyan, Karen, Vinyals, Oriol, Wierstra, Daan, Munos, Rémi, and Silver, David. Learning to search with mctsnets. *CoRR*, abs/1802.04697, 2018.

Guo, Xiaoxiao, Singh, Satinder, Lee, Honglak, Lewis, Richard L, and Wang, Xiaoshi. Deep learning for real-time atari game play using offline monte-carlo tree search planning. In Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N. D., and Weinberger, K. Q. (eds.), *Advances in Neural Information Processing Systems 27*, pp. 3338–3346. Curran Associates, Inc., 2014a.

Guo, Xiaoxiao, Singh, Satinder, Lee, Honglak, Lewis, Richard L, and Wang, Xiaoshi. Deep learning for real-time atari game play using offline monte-carlo tree search planning. In Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N. D., and Weinberger, K. Q. (eds.), *Advances in Neural Information Processing Systems 27*, pp. 3338–3346. Curran Associates, Inc., 2014b.

Ha, David and Schmidhuber, Jürgen. World models. *CoRR*, abs/1803.10122, 2018.

Kalchbrenner, Nal, van den Oord, Aäron, Simonyan, Karen, Danihelka, Ivo, Vinyals, Oriol, Graves, Alex, and Kavukcuoglu, Koray. Video pixel networks. *CoRR*, abs/1610.00527, 2016.

Kalchbrenner, Nal, Elsen, Erich, Simonyan, Karen, Noury, Seb, Casagrande, Norman, Lockhart, Edward, Stimberg, Florian, van den Oord, Aäron, Dieleman, Sander, and Kavukcuoglu, Koray. Efficient neural audio synthesis. *CoRR*, abs/1802.08435, 2018.

Kingma, Diederik P and Ba, Jimmy. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Kingma, Diederik P. and Welling, Max. Auto-encoding variational bayes. *CoRR*, abs/1312.6114, 2013.

Kocsis, Levente and Szepesvári, Csaba. Bandit based monte-carlo planning. In *Proceedings of the 17th European Conference on Machine Learning*, ECML'06, pp. 282–293, Berlin, Heidelberg, 2006. Springer-Verlag. ISBN 3-540-45375-X, 978-3-540-45375-8. doi: 10.1007/11871842_29.

Lavalle, Steven M. Rapidly-exploring random trees: A new tool for path planning. Technical report, 1998.

Lipovetzky, Nir, Ramirez, Miquel, and Geffner, Hector. Classical planning with simulators: Results on the atari video games. In *Proceedings of the 24th International Conference on Artificial Intelligence*, IJCAI'15, pp. 1610–1616. AAAI Press, 2015. ISBN 978-1-57735-738-4.

Luc, Pauline, Neverova, Natalia, Couprie, Camille, Verbeek, Jacob, and LeCun, Yann. Predicting deeper into the future of semantic segmentation. *ICCV*, 2017.

Oh, Junhyuk, Guo, Xiaoxiao, Lee, Honglak, Lewis, Richard, and Singh, Satinder. Action-conditional video prediction using deep networks in atari games. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'15, pp. 2863–2871, Cambridge, MA, USA, 2015. MIT Press.

Pascanu, Razvan, Li, Yujia, Vinyals, Oriol, Heess, Nicolas, Buesing, Lars, Racanière, Sébastien, Reichert, David P., Weber, Theophane, Wierstra, Daan, and Battaglia, Peter. Learning model-based planning from scratch. *CoRR*, abs/1707.06170, 2017.

Pepels, T., Winands, M. H. M., and Lanctot, M. Real-time monte carlo tree search in ms pac-man. *IEEE Transactions on Computational Intelligence and AI in Games*, 6(3):245–257, Sept 2014. ISSN 1943-068X. doi: 10.1109/TCIAIG.2013.2291577.

Ramachandran, Prajit, Paine, Tom Le, Khorrami, Pooya, Babaeizadeh, Mohammad, Chang, Shiyu, Zhang, Yang, Hasegawa-Johnson, Mark A, Campbell, Roy H, and Huang, Thomas S. Fast generation for convolutional autoregressive models. *arXiv preprint arXiv:1704.06001*, 2017.

Rosin, Christopher D. Multi-armed bandits with episode context. *Annals of Mathematics and Artificial Intelligence*, 61(3):203–230, Mar 2011. ISSN 1573-7470. doi: 10.1007/s10472-011-9258-6.

Salimans, Tim, Karpathy, Andrej, Chen, Xi, and Kingma, Diederik P. Pixelcnn++: A pixelcnn implementation with discretized logistic mixture likelihood and other modifications. In *ICLR*, 2017.

Schmidhuber, Jürgen. On learning to think: Algorithmic information theory for novel combinations of reinforcement learning controllers and recurrent neural world models. *CoRR*, abs/1511.09249, 2015.

Silver, David, Huang, Aja, Maddison, Chris J., Guez, Arthur, Sifre, Laurent, van den Driessche, George, Schrittwieser, Julian, Antonoglou, Ioannis, Panneershelvam, Veda, Lanctot, Marc, Dieleman, Sander, Grewe, Dominik, Nham, John, Kalchbrenner, Nal, Sutskever, Ilya, Lillicrap, Timothy, Leach, Madeleine, Kavukcuoglu, Koray, Graepel, Thore, and Hassabis, Demis. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587):484–489, January 2016a. doi: 10.1038/nature16961.

Silver, David, van Hasselt, Hado, Hessel, Matteo, Schaul, Tom, Guez, Arthur, Harley, Tim, Dulac-Arnold, Gabriel, Reichert, David P., Rabinowitz, Neil C., Barreto, André, and Degris, Thomas. The predictron: End-to-end learning and planning. *CoRR*, abs/1612.08810, 2016b.

Silver, David, Schrittwieser, Julian, Simonyan, Karen, Antonoglou, Ioannis, Huang, Aja, Guez, Arthur, Hubert, Thomas, Baker, Lucas, Lai, Matthew, Bolton, Adrian, Chen, Yutian, Lillicrap, Timothy, Hui, Fan, Sifre, Laurent, van den Driessche, George, Graepel, Thore, and Hassabis, Demis. Mastering the game of go without human knowledge. *Nature*, 550:354–, October 2017.

Stentz, Anthony. The focussed d* algorithm for real-time replanning. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence - Volume 2*, IJCAI'95, pp. 1652–1659, San Francisco, CA, USA, 1995. Morgan Kaufmann Publishers Inc. ISBN 1-55860-363-8.

Sukhbaatar, Sainbayar, Lin, Zeming, Kostrikov, Ilya, Synnaeve, Gabriel, Szlam, Arthur, and Fergus, Rob. Intrinsic motivation and automatic curricula via asymmetric self-play. *arXiv preprint arXiv:1703.05407*, 2017.

van den Berg, J., Ferguson, D., and Kuffner, J. Anytime path planning and replanning in dynamic environments. In *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.*, pp. 2366–2371, May 2006. doi: 10.1109/ROBOT.2006.1642056.

van den Oord, Aaron, Kalchbrenner, Nal, Espeholt, Lasse, kavukcuoglu, koray, Vinyals, Oriol, and Graves, Alex. Conditional image generation with pixelcnn decoders. In Lee, D. D., Sugiyama, M., Luxburg, U. V., Guyon, I., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 29*, pp. 4790–4798. Curran Associates, Inc., 2016a.

van den Oord, Aäron, Li, Yazhe, Babuschkin, Igor, Simonyan, Karen, Vinyals, Oriol, Kavukcuoglu, Koray, van den Driessche, George, Lockhart, Edward, Cobo, Luis C., Stimberg, Florian, Casagrande, Norman, Grewe, Dominik, Noury, Seb, Dieleman, Sander, Elsen, Erich, Kalchbrenner, Nal, Zen, Heiga, Graves, Alex, King, Helen, Walters, Tom, Belov, Dan, and Hassabis, Demis. Parallel wavenet: Fast high-fidelity speech synthesis. *CoRR*, abs/1711.10433, 2017a.

van den Oord, Aaron, Vinyals, Oriol, and kavukcuoglu, koray. Neural discrete representation learning. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 30*, pp. 6306–6315. Curran Associates, Inc., 2017b.

van den Oord, Aron, Dieleman, Sander, Zen, Heiga, Simonyan, Karen, Vinyals, Oriol, Graves, Alexander, Kalchbrenner, Nal, Senior, Andrew, and Kavukcuoglu, Koray. Wavenet: A generative model for raw audio. In *Arxiv*, 2016b.

Zhang, Amy, Lerer, Adam, Sukhbaatar, Sainbayar, Fergus, Rob, and Szlam, Arthur. Composable planning with attributes. *CoRR*, abs/1803.00512, 2018.