Unsupervised Creation of Robust Environment Representations for Reinforcement Learning

McKell Stauffer, Maria Fabiano, Sydney Thompson, David Wingate
Brigham Young University

Abstract

Reinforcement learning (RL) methods traditionally have suffered from large sample complexities, which reduces their widespread effectiveness. This is particularly true for the beginning stages of deep RL methods, where the methods must learn what the objects in an environment are in addition to what actions to take. We aim to improve the sample complexity of deep RL methods by disentangling these two tasks, using unsupervised learning methods to complete the first task. In particular, we propose a method that embeds learned environmental representations into a Deep Q-Network (DQN), so that the DQN does not have to learn the environment itself. Specifically, we provide a Rainbow DQN with robust environment representations of Atari video games created through a conditional variational ladder autoencoder with coordinate convolutions.

1 Introduction

Reinforcement learning (RL) is a popular area of current research across various fields, including psychology, neuroscience and computer science [7]. RL methods train autonomous agents to choose actions that maximize an aggregate reward. These methods learn through trial and error and consequently are subject to scalability issues. In particular, they struggle with memory and computational complexity [1].

Even when these algorithms are scalable, they may have a large sample complexity. For example, Lake et al. describe a Deep Q-Network (DQN) that was able to achieve human-level performance on some Atari video games, but it took about 500 times more experience for the algorithm to learn to play the games as compared to human subjects [9]. The drastic difference was even more pronounced in the early training stages as humans had basic knowledge of the video game environments that the DQN did not have.

We aim to improve the sample efficiency of RL algorithms by embedding in them representations of their environment (created through unsupervised learning methods). In particular, we create robust representations of Atari video game states through the use of a conditional variational ladder autoencoder with coordinate convolutions. We then concatenate these compressed representations onto the states in a Rainbow DQN.

The main contributions of this paper are as follows:

1. We create robust representations of the Atari game environment.
2. We improve the sample efficiency of Rainbow.
3. We successfully embed unsupervised learning into reinforcement learning.

2 Preliminaries

2.1 Autoencoders

An *autoencoder* is an unsupervised neural network that attempts to reconstruct its input. The network can be seen as having two parts, an encoder function $f(\cdot)$ and a decoder function $g(\cdot)$. The encoder takes the input $\mathbf{x}$ and embeds it into a compressed space, often referred to as the *latent space representation*, $\mathbf{z} = f(\mathbf{x})$. The decoder takes this representation and attempts to construct the original input $\tilde{\mathbf{x}} = g(\mathbf{z})$. In other words, the goal of the autoencoder is to learn $f(\cdot)$ and $g(\cdot)$ so that $g(f(\mathbf{x})) = \mathbf{x}$. It learns f and g by minimizing a loss function between $\mathbf{x}$ and $\tilde{\mathbf{x}}$. A commonly used loss function for autoencoders is mean-squared error.

A *variational autoencoder* (VAE) forces f to produce a latent representation $\mathbf{z}$ that follows a standard Gaussian distribution [8]. It does this by having the encoder map to a vector of means $\boldsymbol{\mu}$ and vector of standard deviations $\boldsymbol{\sigma}$. It then draws samples from these vectors, resulting in the latent representation $\mathbf{z} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\sigma})$. A latent loss, the Kullback-Liebler (KL) divergence between the latent space and standard Gaussian distributions, is added to the overall loss of the autoencoder.

A *variational ladder autoencoder* (VLAE) defines a joint distribution of latent representations $p(\mathbf{z}_1, ..., \mathbf{z}_L)$, where L is the number of representations [21]. Each $\mathbf{z}_i$ is the resulting encoding of a network f_i. Each f_i has a progressively deeper structure, leading to the varying levels of abstraction for the $\mathbf{z}_i$'s. These networks are constructed in a ladder-like fashion by having the input to each network f_i be the last layer of the previous network (before it maps to $\boldsymbol{\mu}_i$ and $\boldsymbol{\sigma}_i$). In the case of f_1, the network encodes the input image. As the network is variational, each $\mathbf{z}_i$ is drawn from the Gaussian distribution $\mathbf{z}_i \sim \mathcal{N}(\boldsymbol{\mu}_i, \boldsymbol{\sigma}_i)$. The input of the model is then reconstructed using a ladder of decoder networks, where each g_i inputs the concatenation of $\mathbf{z}_i$ and the output of the previous decoder $\tilde{\mathbf{z}}_{i+1}$

$$\tilde{\mathbf{z}}_i = g_i(\mathbf{z}_i \frown \tilde{\mathbf{z}}_{i+1}).$$

In the case of the first decoder g_L, it only takes as input $\mathbf{z}_L$. The last decoder g_1 reconstructs the original image. The loss of the VLAE is the exact same as that of a normal VAE, where the latent representation is defined to be the concatenation of each $\mathbf{z}_i$,

$$\mathbf{z} = \mathbf{z}_1 \frown \cdots \frown \mathbf{z}_i \frown \cdots \frown \mathbf{z}_L.$$

Conditional VAEs (CVAEs) are used to model multi-modal distributions [16]. In particular, they condition specific layers of a VAE on a random variable, often a class. This conditioning of layers on a random variable has been applied to other types of generative networks. As an example of this, the *conditional instance normalization* method was developed for style transfer networks [3]. After normalization, this method shifts and scales each network layer $\mathbf{h}$

$$\mathbf{h} = \boldsymbol{\gamma}_c(\frac{\mathbf{h} - \boldsymbol{\mu}}{\boldsymbol{\sigma}}) + \boldsymbol{\beta}_c$$

with separate shift $\boldsymbol{\beta}$ and scale $\boldsymbol{\gamma}$ parameters for each class c.

Coordinate convolutions (CoordConvs) add coordinate information to convolutions in order to make the transform between high-level spatial latents and pixels easier to learn [10]. It does this by concatenating two channels to a convolution's input. The first channel contains the x-value of each coordinate while the second has the y-value. In other words, the first channel's first row is filled with zeros, the second with ones, etc. While the second channel's first column is filled with zeros, the second with ones, etc. The convolution can then decide whether to use this spatial information, or discard it to preserve translational invariance.

2.2 Q-learning models

Q-learning is a RL method that learns a probability distribution over actions given states, or a policy π. The policy is learned through the optimal action-value function

$$Q^{\pi}(s_t, a_t) = \mathbb{E}_{\pi}[R_t | s_t, a_t], \tag{1}$$

where s_t represents the current state, a_t the proposed action, $R_t = \sum_{k=0}^{\infty} \lambda_{t+1}^k r_{t+k+1}$ is the future discounted cumulative reward, r_t the reward given at time t, and λ_t the discount factor. Models are

episodic in practice, so λ_t is a constant $\lambda \in [0, 1]$ through all time except for after the ending time T, where $\lambda_{t>T} = 0$. R_t is discounted by λ_{t+1} to place heavier interest on more immediate rewards. The *Q-value* is the action a_t that maximizes the result of Equation 1. The optimal action-value function follows an identity called the Bellman Equation

$$Q^{\pi}(s_t, a_t) = \mathbb{E}_{s'}[r_{t+1} + \lambda \max_{a'} Q^{\pi}(s', a')|s_t, a_t]. \tag{2}$$

The expectation is taken over all the possible states to model the states stochastically.

A *Deep Q-Network* (DQN) was the first model to combine neural networks and reinforcement learning [11]. Neural networks are often described as function approximators, and in this case a Q-network is a function approximator for discovering the Q-values. The algorithm uniformly samples a mini-batch of transitions (actions with corresponding states and rewards) from the network's replay memory. Each transition can be seen as a state in a Markov decision process. These states are fed iteratively through a Bellman equation update

$$Q_{i+1}(s_t, a_t) = \mathbb{E}_{s'}[r_{t+1} + \lambda \max_{a'} Q_i(s', a')|s_t, a_t] \tag{3}$$

resulting in the policy at each state π_i converging to the best policy π.

A *Rainbow DQN* is an extension of the DQN that incorporates several different improvements upon the algorithm [6]. These improvements include decomposing the max operation in the target into action selection and action evaluation to reduce overestimation, prioritized experience replay, and using stochastic layers for exploration [19, 15, 5].

3 Related work

Previous research into improving the sample efficiency of RL methods has often involved transfer learning. In their papers, Taylor et al. survey and analyze transfer learning in RL contexts [17, 18]. They find that there has been transfer success in the algorithms learning navigation and robot soccer keepaway. For example, Zhang et al. used RL to teach a robot to navigate a maze [20]. They were able to transfer knowledge from previously mastered mazes to new ones. Another approach was taken by Moreno et al. They attempted to embed prior knowledge into the algorithm by restricting its exploration space [12].

More specific to the learning of Atari games, Parisotto et al. used an actor-critic method to train an agent to simultaneously learn on a variety of tasks and then generalize what it has learned to other tasks [13]. They found that their pre-learning methods led to significant speed up in their deep Q-network learning to play 3/7 Atari games that they tested it on. They attributed the speed up of two of these games (Breakout and Pinball) to the fact that pre-learning was done on a similar game (Pong). Chaplot et al. repeated their results in 3D video game environments [2].

While the above-mentioned research transfers knowledge from one RL method to another trained on a different task, we want to transfer knowledge from an unsupervised method. Although there has been recent research in combining unsupervised and RL methods, it focuses mainly on methods in the two fields that could be deemed equivalent. For example, Pfau and Vinyals successfully used a generative adversarial network (GAN) as an actor-critic method where the actor cannot effect the reward [14]. Similarly, Fin et al. showed that GANs and some Inverse RL methods are equivalent [4]. In contrast to these papers combining unsupervised and reinforcement learning by showing their equivalence in certain situations, we aim to use them in tandem, including their inherent differences.

4 Methods

4.1 Conditional VLAE with CoordConv

To create our learned representations, we used a VLAE. We decided to use this model as Zhao et al. showed that it excels at finding disentangled representations [21]. Additionally, we wanted to use its capabilities to model various levels of abstraction. As we are testing in an Atari video game environment, we made the VLAE conditional to better model the different games. We made it conditional by using conditional instance normalization on each layer in the VLAE. Additionally, we replaced each convolutional layer with a CoordConv layer. We did this to lessen the mode collapse of the placement of objects in the VLAE reconstructions.

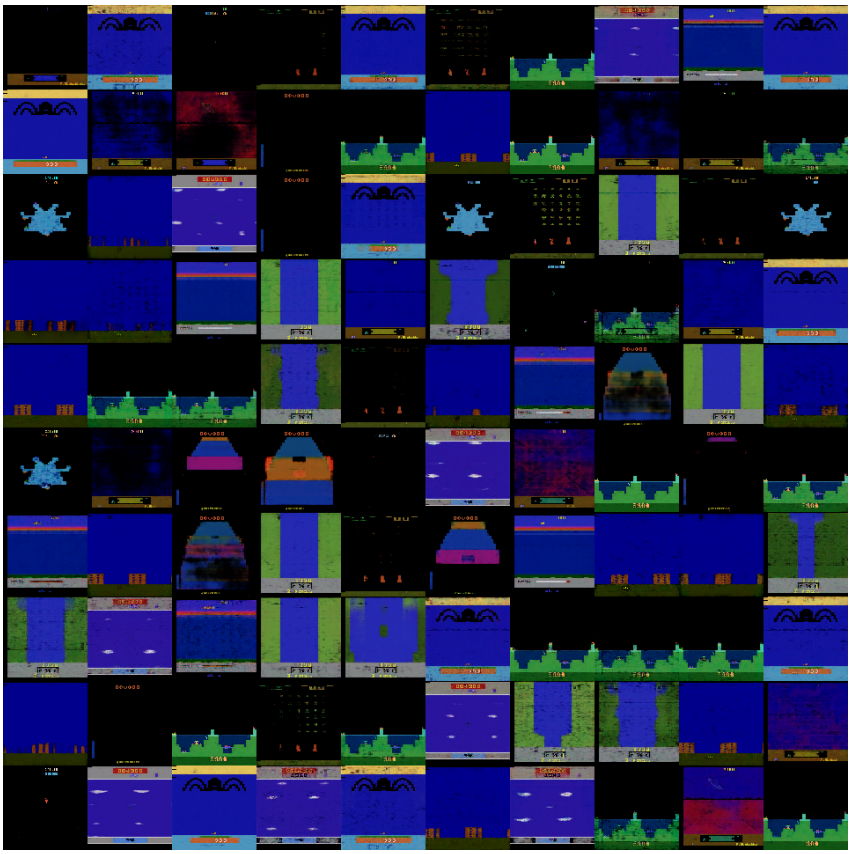

Figure 1: Random reconstructions of Atari games using the trained VLAE.

We trained the VLAE on randomized screenshots from 10 Atari games,[1] obtained by simulating their respective video game environments and taking random actions. Random reconstructions of the VLAE can be seen in Figure 1. Parameters for our implementation, along with ablation studies on the VLAE can be found in Appendix A.

4.2 Rainbow evaluation

We incorporate the VLAE into Rainbow when it is feeding the transitions (states, actions, and rewards) into the replay memory. Specifically, we embed each state through the VLAE and concatenate it to the bottom of the state. When testing on a class that the VLAE has never seen, we pass in a random class into the conditional invariance normalization. We hypothesize that this added information about the object representations will help Rainbow to be more sample efficient in the early stages.

5 Results

We ran Rainbow with the VLAE embeddings on Atlantis, Name This Game, Seaquest, Space Invaders and Chopper Command. Notice that we did not train the VLAE on Chopper Command. Figures 2-6 compare the average reward (taken over 10 game simulations) of Rainbow when it has access to the VLAE embeddings and when it does not.

We find that our method had a higher average reward in two of the games (Space Invaders and Atlantis), a similar reward in two games (Name This Game and Chopper Command) and worse in one of the games (Seaquest). We also note that the reward for Rainbow with the VLAE embeddings has much higher variance than for the original Rainbow implementation.

[1] Air Raid, Atlantis, Gravitar, Name This Game, River Raid, Seaquest, Solaris, Space Invaders, Time Pilot, and Zaxxon.

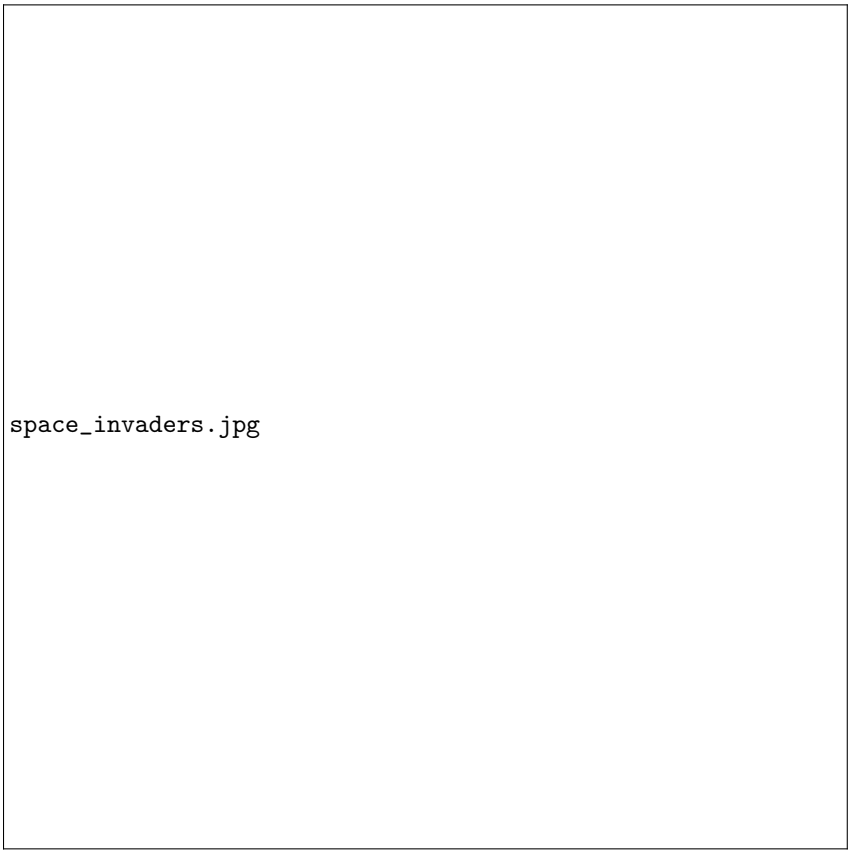

Figure 2: The average reward (taken over 10 game simulations) of Rainbow with and without the VLAE embeddings for Space Invaders.

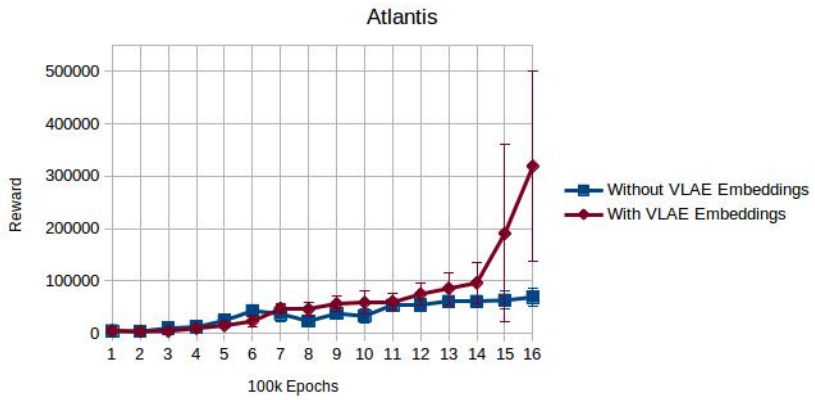

Figure 3: The average reward (taken over 10 game simulations) of Rainbow with and without the VLAE embeddings for Atlantis.

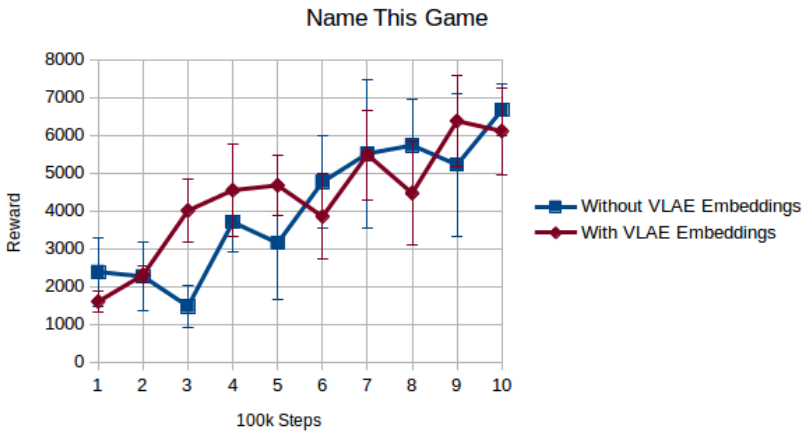

Figure 4: The average reward (taken over 10 game simulations) of Rainbow with and without the VLAE embeddings for Name This Game.

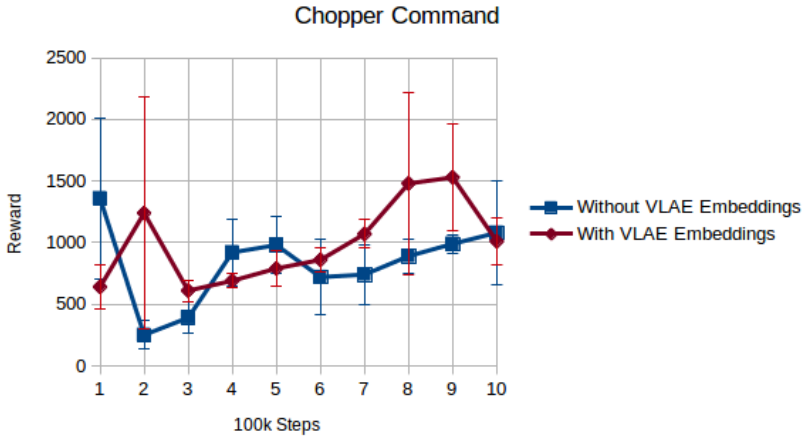

Figure 5: The average reward (taken over 10 game simulations) of Rainbow with and without the VLAE embeddings for Chopper Command.

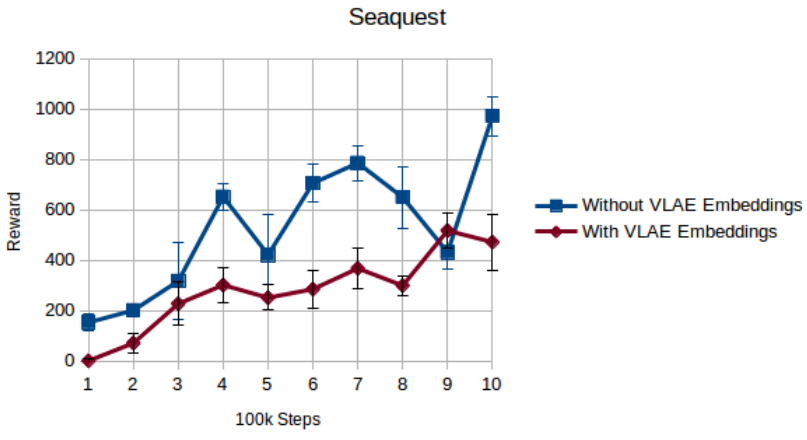

Figure 6: The average reward (taken over 10 game simulations) of Rainbow with and without the VLAE embeddings for Seaquest.

6 Conclusion

We found that in a few promising cases, a conditional variational ladder autoencoder with coordinate convolution learned objects representations that decreased the sample complexity of the Rainbow DQN, especially at early stages of training. Future work includes:

1. Actually including results for using the conditional instance normalization...
2. Embeddings four images a time to catch dynamics
3. Trying other unsupervised and reinforcement learning algorithms

References

[1] Kai Arulkumaran, Marc Peter Deisenroth, Miles Brundage, and Anil Anthony Bharath. Deep reinforcement learning: A brief survey. *IEEE Signal Processing Magazine*, 34(6):26–38, November 2017.

[2] Devendra Singh Chaplot, Guillaume Lample, Kanthashree Mysore Sathyendra, and Ruslan Salakhutdinov. Transfer deep reinforcement learning in 3d environments: An empirical study. In *Proceedings of the 30th Conference on Neural Information Processing Systems (NIPS)*, Barcelona, Spain, 2016.

[3] Vincent Dumoulin, Jonathan Shlens, and Manjunath Kudlur. A learned representation for artistic style. In *Proceedings of 5th International Conference on Learning Representations (ICLR)*, 2017.

[4] Chelsea Finn, Paul Christiano, Pieter Abbeel, and Sergey Levine. A connection between generative adversarial networks, inverse reinforcement learning, and energy-based models. *CoRR*, abs/1611.03852, November 2016.

[5] Meire Fortunato, Mohammad Gheshlaghi Axar, Bilal Piot, Jacob Menick, Ian Osband, Alex Graves, Vlad Mnih, Remi Munos, Demis Hassabis, Olivier Pietquin, Charles Blundell, and Shane Legg. Noisy networks for exploration. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2018.

[6] Matteo Hessel, Joseph Modayil, Hado van Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney, Daniel Horgan, Bilal Piot, Mohammad Gheshlaghi Azar, and David Silver. Rainbow: Combining improvements in deep reinforcement learning. *CoRR*, abs/1710.02298, 2017.

[7] Leslie Pack Kaelbling, Michael L. Littman, and Andrew W. Moore. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, 4(1):237–285, January 1996.

[8] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. In *Proceedings of 2nd International Conference on Learning Representations (ICLR)*, 2014.

[9] Brenden M. Lake, Tomer D. Ullman, Joshua B. Tenenbaum, and Samuel J. Gershman. Building machines that learn and think like people. *Behavioral and Brain Sciences*, 40:e253, 2017.

[10] Rosanne Liu, Joel Lehman, Piero Molino, Felipe Petroski Such, Eric Frank, Alex Sergeev, and Jason Yosinski. An intriguing failing of convolutional neural networks and the coordconv solution. In *Proceedings of 32nd Annual Conference on Neural Information Processing Systems (NeurIPS)*, 2018.

[11] Volodymyr Mnih, Koray Kavukcuoglu, Silver David, Andrei A Ruso, 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. *Nature*, 518:529–533, February 2015.

[12] David L. Moreno, Carlos V. Regueiro, Roberto Iglesias, and Senén Barro. Using prior knowledge to improve reinforcement learning in mobile robotics. In *Proceedings of the Towards Autonomous Robotics Systems (TAROS)*, UK, 2004.

[13] Emilio Parisotto, Jimmy Ba, and Ruslan Salakhutdinov. Actor-mimic: Deep multitask and transfer reinforcement learning. In *Proceedings of the International Conference on Learning Representations (ICLR)*, Puerto Rico, 2016.

[14] David Pfau and Oriol Vinyals. Connecting generative adversarial networks and actor-critic methods. In *Proceedings of the 30th Neural Information Processing Systems (NIPS) Workshop on Adversarial Training*, Barcelona, Spain, 2016.

[15] Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2016.

[16] Kihyuk Sohn, Xinchen Yan, and Honglak Lee. Learning structured output representation using deep conditional generative models. In *Proceedings of Advances in Neural Information Processing Systems 28 (NIPS 2015)*, 2015.

[17] Matthew E. Taylor and Peter Stone. Transfer learning for reinforcement learning domains: A survey. *Journal of Machine Learning Research*, 10:1633–1685, July 2009.

[18] Matthew E. Taylor and Peter Stone. An introduction to intertask transfer for reinforcement learning. *AI Magazine*, 32(1):15–34, March 2011.

[19] Hado van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence (AAAI-16)*, pages 2094–2100, 2016.

[20] Jingwei Zhang, Jost Tobias Springenberg, Joschka Boedecker, and Wolfram Burgard. Deep reinforcement learning with successor features for navigation across similar environments. *CoRR*, abs/1612.05533, December 2016.

[21] Shengjia Zhao, Jiaming Song, and Stefano Ermon. Learning hierarchical features from deep generative models. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 4091–4099, International Convention Centre, Sydney, Australia, August 2017.

Table 1: Parameters for the VLAE implementation

Parameter Description	Parameter Value
Dimension of post-processing images	[96, 96, 3]
Range of post-processing image values	[-1.0, 1.0]
Batch size	100
Number of layers in encoder	4
Learning rate	0.0002
Dimension of each latent	[21]
Number of outputs of each layer	{64, 128, 256, 512, 1024}
Kernel size	[4, 4]
Stride	{1, 2}

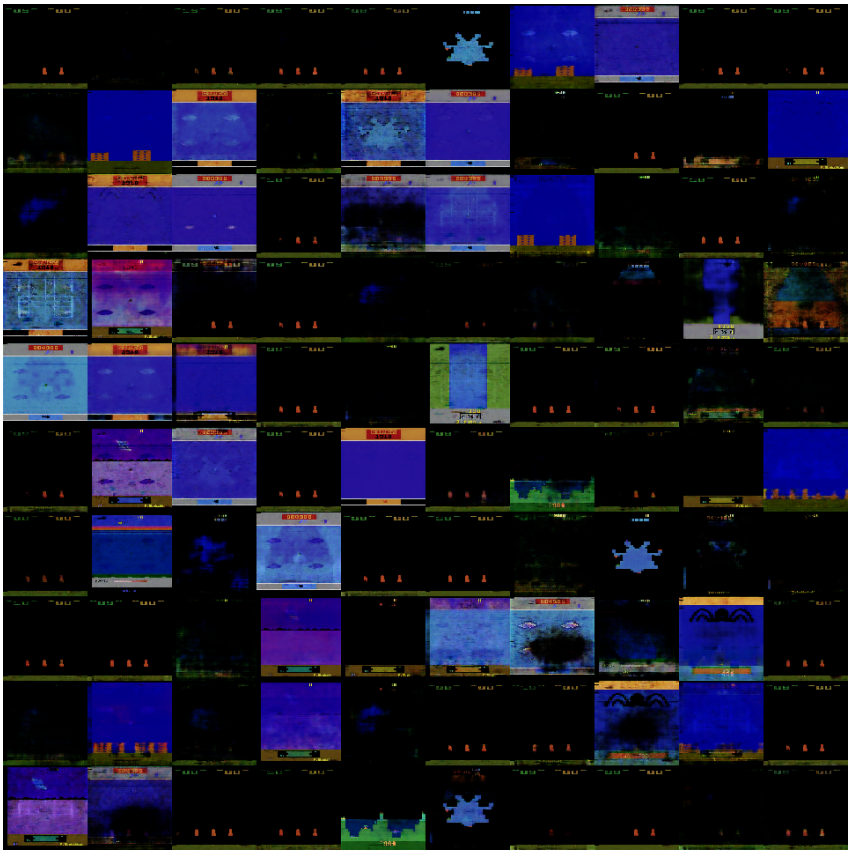

Figure 7: Random reconstructions of Atari games using the trained VLAE without conditional instance normalization. In these reconstructions, one could see mode collapse and mixing up of the games. Compare to Figure 1.

A Implementation and Ablation Studies

Table 1 contains the parameters for the VLAE implementation. Table 2 contains the parameters for the Rainbow implementation. As for the ablation experiments, Figures 7 & 8 are the reconstructions without conditioning and coordinate convolutions respectively.

Table 2: Parameters for the Rainbow implementation

Parameter Description	Parameter Value
Max episode length	108,000
Number of states processed at a time	4
Network hidden size	512
Initial standard deviation of noisy layers	0.1
Discretized size of value distribution	51
Replay memory capacity	1,000,000
Prioritized experience replay exponent	0.5
Minimum of value distribution	-10
Maximum of value distribution	10
Frequency of sampling from memory	4
Initial prioritized experience replay importance sampling weight	0.4
Number of steps for multi-step return	3
Discount	0.99
Number of steps after which to update target network	32,000
Reward clipping	1
Learning rate	0.0000625
Batch size	32
Number of steps before training starts	80,000
Number of evaluation episodes to average over	10
Number of transitions to use for validating Q	500

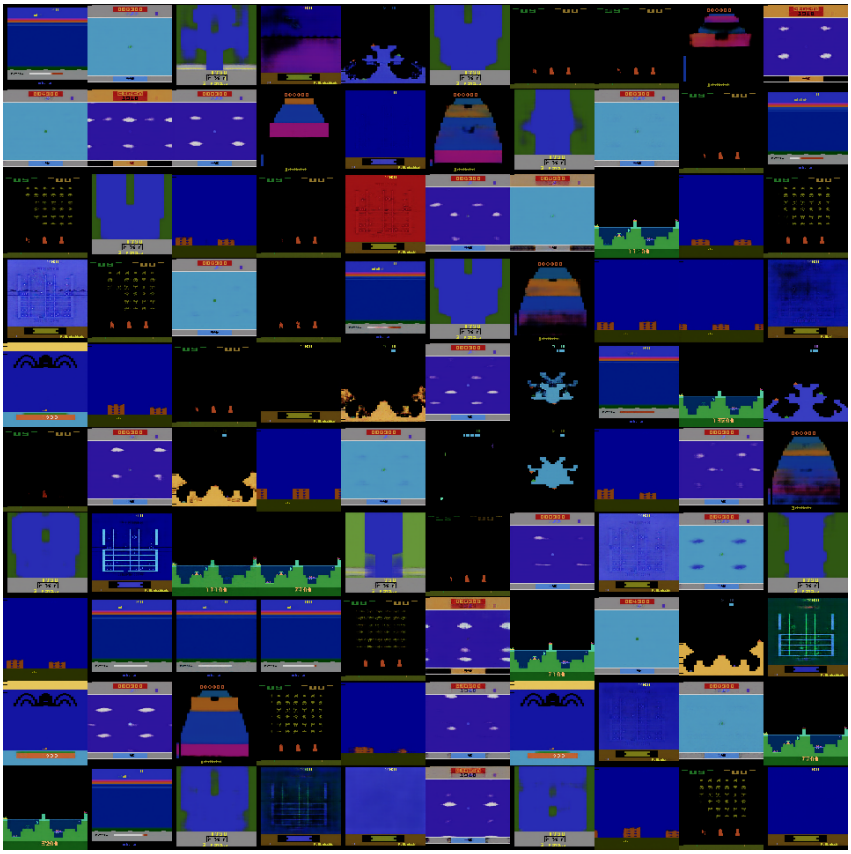

Figure 8: Random reconstructions of Atari games using the trained VLAE without coordinate convolutions. In these reconstructions, one could see mode collapse (the backgrounds are there, but no ships or other changing details). Compare to Figure 1.