Received: 16 May 2019 | Revised: 10 October 2019 | Accepted: 24 January 2020

DOI: 10.1111/exsy.12537

SPECIAL ISSUE PAPER

Expert Systems WILEY

Increasing sample efficiency in deep reinforcement learning using generative environment modelling

Per-Arne Andersen | Morten Goodwin | Ole-Christoffer Granmo

Department of ICT, University of Agder, Grimstad, Norway

Correspondence
Per-Arne Andersen, Jon Lilletunsvei 9, 4879 Grimstad, Norway.
Email: per.andersen@uia.no

Abstract

Reinforcement learning is a broad scheme of learning algorithms that, in recent times, has shown astonishing performance in controlling agents in environments presented as Markov decision processes. There are several unsolved problems in current state-of-the-art that causes algorithms to learn suboptimal policies, or even diverge and collapse completely. Parts of the solution to address these issues may be related to short- and long-term planning, memory management and exploration for reinforcement learning algorithms. Games are frequently used to benchmark reinforcement learning algorithms as they provide a flexible, reproducible and easy to control environments. Regardless, few games feature the ability to perceive how the algorithm performs exploration, memorization and planning. This article presents *The Dreaming Variational Autoencoder with Stochastic Weight Averaging and Generative Adversarial Networks* (DVAE-SWAGAN), a neural network-based generative modelling architecture for exploration in environments with sparse feedback. We present deep maze, a novel and flexible maze game-engine that challenges DVAE-SWAGAN in partial and fully observable state-spaces, long-horizon tasks and deterministic and stochastic problems. We show results between different variants of the algorithm and encourage future study in reinforcement learning driven by generative exploration.

KEYWORDS

artificial experience-replay, deep reinforcement learning, environment Modelling, exploration, generative adversarial networks, generative Modelling, Markov decision processes, model-based RL, neural networks, variational autoencoder

1 | INTRODUCTION

Reinforcement learning (RL) is a field of research that has quickly become one of the most promising branches of machine learning algorithms to solve artificial general intelligence (Arulkumaran, Deisenroth, Brundage and Bharath 2017; Kaelbling, Littman and Moore 1996; Li 2017 and Mousavi, Schukat and Howley 2018). There have been several breakthroughs in reinforcement learning research in recent years for

Extension of prior study found in Andersen, Goodwin and Granmo (2017, 2018).

environments such as the Atari Arcade (Mnih *et al*. 2013, 2015), AlphaZero (Silver *et al*. 2017), OpenAI Five and AlphaStar (Arulkumaran, Cully and Togelius 2019), but no algorithms are capable of human performance without extensive hardware requirements that are accessible to only a few institutions, such as Deep Mind and Open AI. Due to this, reinforcement learning still has several open research questions to address before the general public can deploy these algorithms successfully. It is possible that many of the issues in the current state of the art can be resolved with algorithms that adequately accounts for planning, exploration and memory efficiency at different time-horizons.

In current state-of-the-art RL algorithms, long-horizon RL tasks are difficult to master because there is as of yet no optimal exploration algorithm that is capable of proper state-space pruning. Exploration strategies such as ϵ-greedy are widely used in RL, but cannot find an adequate exploration/exploitation balance without exhaustive hyperparameter-tuning. Environment modelling is a promising exploration method where the goal is to imitate the behaviour of a target environment. By constructing an artificial model of the environment, the need to interact with the ground-truth environment is reduced significantly. This enables the RL-algorithm to explore large parts of the state space without the cost of exhausting the ground-truth environment. A balance between exploration and exploitation must be accounted for to learn the underlying dynamics of the environment. The algorithm must, therefore, select observations carefully to only learn essential features during the exploration phase.

By combining generative modelling with deep reinforcement learning, we find that it is possible for agents to learn optimal policies using only generated training data samples. The approach that we present is the dreaming variational autoencoder with two extensions, based on generative adversarial networks and stochastic weight averaging. We also present a new learning environment, deep maze, that aims to bring a vast set of challenges for reinforcement learning algorithms and is the environment used for testing the presented algorithms.

This article is organized as follows. Section 2 surveys recent study in the field. Section 3 briefly introduces the reader to preliminaries. Section 4 proposes *The Dreaming Variational Autoencoder* for environment modelling to improve exploration in RL. Section 5 introduces the deep maze learning environment for exploration, planning and memory management research for reinforcement learning. Section 6 shows results in the deep line wars environment and that RL agents can be trained to navigate through the deep maze environment using only artificial training data. Finally, we summarize our findings and outlines future study in Section 7.

2 | LITERATURE REVIEW

In machine learning, the goal is to create an algorithm that is capable of accurately representing a target environment. There is, however, little literature on generative modelling for game environments on the scale we propose in this article. The primary focus of recent RL research has been through improvements in on and off policy algorithms, while less attention has been put into generative exploration. This section introduces a thorough literature review of reinforcement-based generative modelling.

Bangaru, Suhas and Ravindran (2016) proposed a method of deducing the Markov Decision Process (MDP) by introducing an adaptive exploration signal (pseudo-reward), which was obtained using deep generative model. Their approach was to compute the Jacobian of each state and used it as the pseudo-reward when using deep neural networks to learn the state-generalization.

Xiao and Kesineni (2016) proposed the use of generative adversarial networks (GAN) for model-based reinforcement learning. The goal was to utilize GAN for learning dynamics of the environment in a short-horizon timespan and combine this with the strength of far-horizon value iteration RL algorithms. The GAN architecture proposed illustrated near authentic generated images giving comparable results to Mnih *et al*. (2013).

Higgins *et al*. (2017) proposed DARLA, an architecture for modelling the environment using β-VAE (Higgins *et al*. 2016). The trained model was used to learn the optimal policy of the environment using algorithms such as Deep Q-Networks (DQN) (Mnih *et al*. 2015), Asynchronous Actor-Critic Agents (A3C) (Mnih *et al*. 2016) and Episodic Control (Blundell *et al*. 2016). DARLA is to the best of our knowledge, the first algorithm to introduce learning without access to the ground-truth environment during training.

Buesing *et al*. (2018) recently compared several methods of environment modelling, showing that it is far better to model the state-space then to utilize Monte-Carlo rollouts (RAR). The proposed architecture, state-space models (SSM) was significantly faster and produced acceptable results compared to auto-regressive (AR) methods.

The algorithm VMAV-C is a combination of VAE and attention-based value function (AVF), and mixture density network recurrent neural network (MDN-RNN) from Ha and Schmidhuber (2018). This modification to the original World Models algorithm improved performance in the Cart Pole environment. They used the on-policy algorithm PPO to learn the optimal policy from the latent representation of the state-space (Liang, Wang, Feng, Liu and Huang 2018).

Deep Planning Network (PlaNet) is a model-based agent that interpret the pixels of a state to learn the dynamics of an environment. The environment dynamics are stored into latent-space, where the agent sample actions based on the learned representation. The proposed algorithm showed significantly better sample efficiency compared to algorithms such as A3C (Hafner *et al*. 2018).

Chua, Calandra, McAllister and Levine (2018) recently proposed Probabilistic Ensembles with Trajectory Sampling (PETS). The algorithm uses an ensemble of bootstrap neural networks to learn a dynamics model of the environment over future states. The algorithm then uses this model to predict the best action for future states. The authors show that the algorithm significantly lowers sampling requirements for environments such as half-cheetah compared to SAC and PPO.

Stochastic optimal control with latent representations (SOLAR) is an algorithm that learns the dynamics of the environment by exploiting the knowledge from a reinforcement learning policy. This enables the algorithm to learn local models which are used in policy learning for complex systems. SOLAR is built around a probabilistic graphical model (PGM) structure that allows efficient learning of the environment model. By exploiting the locality of the model, the authors show that the gradients give good direction for policy improvements during training. The algorithm was compared to model-free methods and showed significantly better performance and data efficient compared to algorithms such as LQR-FLM. (Zhang, Patras and Haddadi 2018)

Ha and Schmidhuber (2018) proposed in *Recurrent World Models Facilitate Policy Evolution*, a novel architecture for training RL algorithms using variational autoencoders. This article showed that agents could successfully learn the environment dynamics and use this as an exploration technique requiring no interaction with the target domain. The architecture is mainly three components; vision, controller and model, the vision model is a variational autoencoder that outputs a latent-space variable of an observation. The latent-space variable is processed in the model and is fed into the controller for action decisions. Their algorithms show state-of-the-art performance in self-supervised generative modelling for reinforcement learning agents.

One of the most recent advancements in generative modelling for reinforcement learning is the Neural Differential Information Gain Optimisation (NDIGO) algorithm by Azar *et al*. (2019), a self-supervised exploration model that learns a world model representation from noisy data. The primary features of NDIGO are its robustness to noise due to their method to cancel out negative loss and to give positive learning more value. The authors show in their maze environment that the model successfully converges towards an optimal world model even when introducing noise. The author claims that the algorithm outperforms previous state-of-the-art, being the Recurrent World Model from.

3 | BACKGROUND

We base our work on the well-established theory of reinforcement learning originally formulated in Sutton, Precup and Singh (1999), defining the problem as a MDP as $\langle \mathcal{S}, \mathcal{A}, \mathcal{R}, \mathcal{T} \rangle$ pairs. The state-space, $\mathcal{S}$ represents all possible states while the action-space, $\mathcal{A}$ represents all available actions the agent can perform in the environment. $\mathcal{R}$ is the reward function while $\mathcal{T}$ denotes the transition function $(\mathcal{T} : \mathcal{S} \times \mathcal{A} \to \mathcal{S})$, which is a mapping from state $s_t \in \mathcal{S}$ and action $a_t \in \mathcal{A}$ to the future state s_{t+1}. After each performed action, the environment dispatches a reward signal, $\mathcal{R} : \mathcal{S} \to r$.

We call a sequence of states and actions a *trajectory* denoted as $\tau = (s_0, a_0, \ldots, s_t, a_t)$ and the sequence is sampled through the use of a stochastic policy that predicts the optimal action in any state: $\pi_\theta(a_t | s_t)$, where π is the policy and θ are the parameters. The primary goal of the reinforcement learning is to *reinforce* good behaviour. The algorithm should try to learn the policy that maximizes the total expected discounted reward given by, $\mathcal{J}(\pi) = \mathop{\mathbb{E}}_{(s_t, a_t) \tilde{p}(\pi)} \left[\sum_{j=0}^{T} \gamma^j \mathcal{R}(s_j) \right]$ Mnih *et al*. (2015).

Several algorithms try to address the problem of reinforcement learning, commonly divided into three categories; *Value Iteration*, *Policy Iteration and Actor-Critic Algorithms*, with variations of *on-policy* and *off-policy* learning.

Autoencoders are commonly used in supervised learning to encode arbitrary input to a compact representation, and using a decoder to reconstruct the original data from the encoding. The purpose of autoencoders is to store redundant data into a densely packed vector form. In its simplest form, an autoencoder consists of feed-forward neural network where the input and output layer is of equal neuron capacity and the hidden layer smaller, used to compress the data. Such autoencoder can be defined as: $\phi : \mathcal{X} \to \mathcal{Z}$, $\psi : \mathcal{Z} \to \mathcal{X}$, where $\phi, \psi : \mathrm{argmin}_{\phi,\psi} \left\| \mathcal{X} - (\phi \times \psi) \hat{\mathcal{X}} \right\|^2$. In this

TABLE 1 A comparison of features in general reinforcement learning, variational autoencoders and generative adversarial networks

Algorithm branch	Sample efficient	Sequential data	Generative modelling	Labelled data
Reinforcement learning	No	Yes	No	No
Variational autoencoders	Yes	Yes	Yes	Yes
Generative adversarial networks	Yes	No	Yes	Yes
DVAE-GAN	Yes	Yes	Yes	Yes

Note: The purpose of this illustration is to show that the proposed algorithm uses generative modelling to inherit sample efficient generative modelling for sequential data for use in reinforcement learning algorithms.

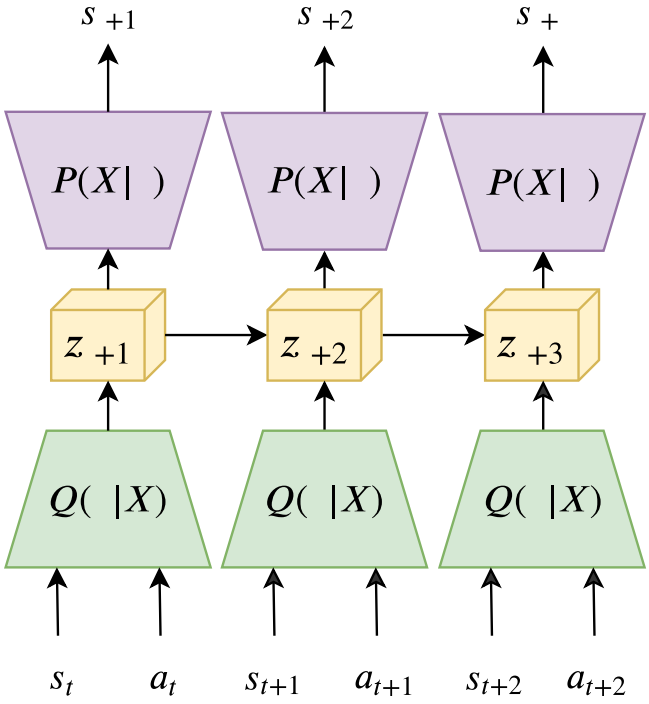


FIGURE 1 Illustration of the DVAE model. The model consumes state and action pairs, yielding the input encoded in latent-space. Latent-space can then be decoded to a probable future state. $\mathcal{Q}(z|X)$ is the encoder, z_t is latent-space and $\mathcal{P}(X|z)$ is the decoder. DVAE can also use LSTM to better learn longer sequences in continuous state-spaces

notation, $\mathcal{X}$ defines the input, $\mathcal{Z}$, the encoding and $\hat{\mathcal{X}}$ as the reconstructed data. There are, however several issues with autoencoders, as they are not generative, in the sense that they can transition between features in the input data. To remedy this, Kingma and Welling (2013) proposed the Variational Autoencoder (VAE) algorithm that enables interpolation between features in the latent space. The interpolation is done by introducing a vector of means μ and a vector of standard deviations σ. These vectors, along with the additional KL-Loss gives the algorithm the ability to learn variations in the input data, enabling the output to be vastly more diverse.

Our background for choosing the following branch of algorithms are described in Table 1. Algorithms based on reinforcement learning learns by hands-on interaction. Experience is attained by exploring the environment, but in some cases, the environment may be exhausted before the agent can learn to behave optimally. To remedy this, we introduce a model-based exploration approach using VAE to model the environment with sequential data as input. This increase the sampling efficiency of the RL algorithm by a significant amount. To increase the performance of the environment model, GAN is used to strengthen the quality of the environment-model, where the VAE and GAN continually reinforce each other's performance.

4 | THE DREAMING VARIATIONAL AUTOENCODER

The Dreaming Variational Autoencoder (DVAE) is an end-to-end solution for generating probable future states $\hat{s}_{t+n}$ from an arbitrary state-space $\mathcal{S}$ using state-action pairs explored prior to s_{t+n} and a_{t+n}. Figure 1 illustrates the DVAE model, where Algorithm 1 works as follows. First, the agent collects experiences for utilizing experience-replay in the *Run-Agent* function. At this stage, the agent explores the state-space guided by a Gaussian distributed policy. The agent acts, observes, and stores the observations into the experience-replay buffer $\mathcal{D}$. After the agent reaches terminal state, the DVAE algorithm encodes state-action pairs from the replay-buffer D into probable future states. This is stored in the replay-buffer for artificial future-states $\hat{D}$.

Algorithm 1 The Dreaming Variational Autoencoder

1: Initialize replay memory $\mathcal{D}$ and $\hat{\mathcal{D}}$ to capacity $\mathcal{N}$
2: Initialize policy π_θ
3: **function** Run-Agent($\mathcal{T}$, $\mathcal{D}$)
4: **for** $i = 0$ to N_EPOCHS **do**
5: Observe starting state, $s_0 \tilde{\mathcal{N}}(0,1)$
6: **while** s_t not TERMINAL **do**
7: $a_t \leftarrow \pi_\theta(s_t = s)$

8: $s_{t+1}, r_t, terminal_t \leftarrow \mathcal{T}(s_t, a_t)$
9: Store experience into replay buffer $\mathcal{D}(s_t, a_t, r_t, s_{t+1}, terminal_t)$
10: $s_t \leftarrow s_{t+1}$
11: **end while**
12: **end for**
13: **end function**
14: Initialize encoder $\mathcal{Q}(z|X)$
15: Initialize decoder $\mathcal{P}(X|z)$
16: Initialize DVAE model $\mathcal{T}_\theta = \mathcal{P}(X|\mathcal{Q}(z|X))$
17: **function** DVAE.
18: **for** d_i in D **do**
19: $s_t, a_t, r_t, s_{t+1} \leftarrow d_i$ ▷ Expand replay buffer pair
20: $X_t \leftarrow s_t, a_t$
21: $z_t \leftarrow \mathcal{Q}(X_t)$ ▷ Encode X_t into latent-space
22: $\hat{s}_{t+1} \leftarrow \mathcal{P}(z_t)$ ▷ Decode z_t into probable future state
23: Store experience into artificial replay buffer $\hat{\mathcal{D}}(\hat{s}_t, a_t, r_t, \hat{s}_{t+1}, terminal_t)$
24: $\hat{s}_t = \hat{s}_{t+1}$
25: **end for**
26: **return** $\hat{\mathcal{D}}$
27: **end function**

Figure 2 illustrates how the algorithm can generate sequences of artificial trajectories using $\mathcal{T}_\theta = \mathcal{P}(X|\mathcal{Q}(z|X))$, where $z = \mathcal{Q}(z|X)$ is the encoder, and $\mathcal{T}_\theta = \mathcal{P}(X|z)$ is the decoder. With state s_0 and action $\mathcal{A}_{right}$ as input, the algorithm generates state $\hat{s}_1$ which in the table can be observed is similar to the real state s_1. With the next input, $\mathcal{A}_{down}$, the DVAE algorithm generates the next state $\hat{s}_2$ which again can be observed to be equal to s_2. Note that this is without ever observing state s_1. Hence, the DVAE algorithm needs to be initiated with a state, for example, s_0, and actions follows. It then generates (dreams) next states.

The requirement is that the environment must be partially discovered so that the algorithm can learn to behave similarly to the target environment. To predict a trajectory of three timesteps, the algorithm does nesting to generate the whole sequence: $\tau = \hat{s}_1, a_1, \hat{s}_2, a_2, \hat{s}_3, a_3 = \mathcal{T}_\theta(\mathcal{T}_\theta(\mathcal{T}_\theta(s_0, \mathcal{A}_{rnd}), \mathcal{A}_{rnd}), \mathcal{A}_{rnd})$. The algorithm does this well early on, but have difficulties with longer state sequences in continuous state-spaces.

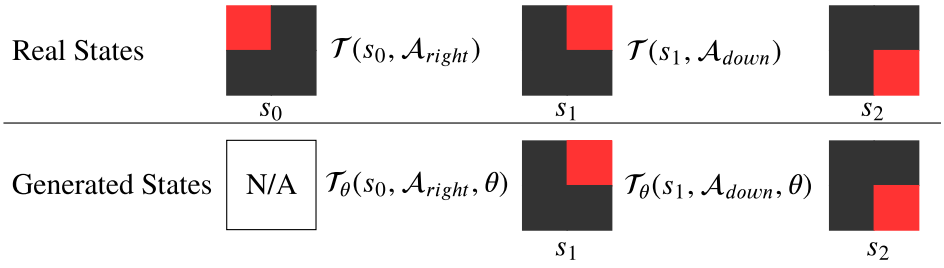


FIGURE 2 DVAE algorithm for generating states using $\mathcal{T}_\theta$ versus the real transition function $\mathcal{T}$. First, a real state is collected from the replay-memory. DVAE can then produce new states from current the trajectory τ using the state-action pairs. θ represent the trainable model parameters

TABLE 2 Feature overview of VAE, GAN, DVAE-GAN and DVAE-SWAGAN

Model	Features	Image quality	Data generalization
VAE	·Converge to local minimum·Stable	·Few artefacts·Blurry	·Low
GAN	·Converge to saddle points·Unstable	·Artefacts·Sharp	·Medium
DVAE-GAN	·Converge to saddle points·Unstable	·Few artefacts·Sharp	·Medium
DVAE-SWAGAN	·Converge to saddle points·Stable	·Few artefacts·Sharp	·High

Note: The highlighted text outlines the benefits of the model.

4.1 | Generative adversarial network approach (DVAE-GAN)

To combat the divergence behaviour in continuous state-space, we extend the model to use generative adversarial networks. The most common cases of divergence are found where there are (a) complex state-transitions, (b) many state-transitions and (c) sparse state-transitions. In general, we find these properties in continuous and stochastic environments. By using an adversarial approach, we see that DVAE-GAN better generalize for such problems and is far more stable due to increased diversity compared to DVAE (See Table 2 for a detailed comparison).

Figure 3 illustrates the proposed DVAE-GAN extension to the original DVAE architecture. In this model, two new components; the *generator G* and *discriminator D* is introduced. This is an adversarial approach previously proposed by Makhzani, Shlens, Jaitly, Goodfellow and Frey (2015). The generator $G(n|\ S_t, A_t)$ samples from a Gaussian distribution, while being conditioned on the current state and action, to predict the latent-space distribution z_{gan}. The discriminator $D(z_{vae}, z_{gan})$ is a neural network that tries to predict the validity of the input, in this case, if the latent-space variable is from the ground-truth distribution. A min-max game between the generator and the discriminator fuels learning where the generator minimizes its error towards the real latent-space and the discriminator learns to distinguish between the real and fake latent distribution. In the VAE model, the latent-space distribution is sampled from $z_{vae} = \alpha_t + (\mu_t \times N(0, 1))$ as proposed by Kingma and Welling (2013). The discriminator should then evaluate z_{vae} to be genuine, and its parameters are updated according to the confidence of the prediction. To increase the stability of the VAE, we add the loss of the discriminator to the VAE loss, where we use the original loss function first proposed by Kingma and Welling (2013).

4.2 | Stochastic weight averaging approach (DVAE-SWAGAN)

We introduced DVAE-GAN which tries to combat divergence for complex environments. The GAN architecture increases the diversity of the latent-space, but model collapse and instability become a problem. Table 2 outlines a quick overview of the benefits of each of the introduced generative models. VAE is known for its stability but has limited capabilities in its latent-space capacity. The quality of VAE is good but suffers from a severe blurring of the decoded latent-space. Compared to other algorithms, the VAE architecture does not generalize well to a diverse data set. The generative adversarial networks have gotten increased attention due to their diverse latent-space. The images of GANs are known for its sharpness, but they suffer from artefacts in the produced output. These networks are state-of-the-art in the sense that they generalize well to the data set. DVAE-GAN as proposed by Makhzani *et al*. (2015) have few artefacts, which is an improvement from the regular GAN architecture. There is, however, high variance in the model which makes it unstable beyond what is seen in vanilla VAE and GAN. To improve the model instability, we introduce Stochastic weight averaging (SWA), first proposed by Izmailov, Podoprikhin, Garipov, Vetrov and Wilson (2018). DVAE-SWAGAN is a

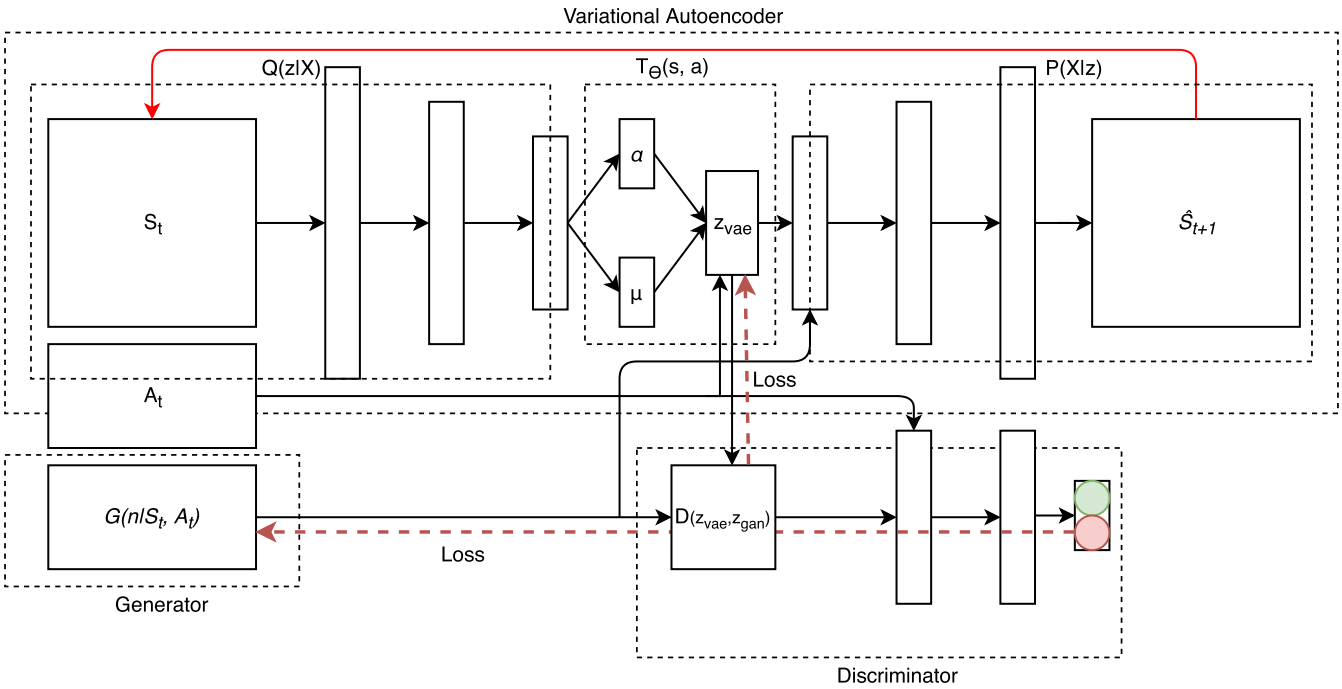


FIGURE 3 The proposed DVAE-GAN architecture. While the original VAE architecture persist, as described in Algorithm 1, a new generative adversarial networks component is added for increased generalization across the latent-space

combination of VAE, SWA and GAN to significantly reduce the variance of the predictions. The algorithm is in principle the averaged ensemble of DVAE-GAN along the trajectory of SGD. We use a cyclical learning rate (Smith 2015) and average the weights each training iteration creating the DVAE-SWAGAN model, seen in Figure 4.

5 | ENVIRONMENTS

The DVAE algorithm was tested in two environments. The first environment is the deep line wars that were introduced by Andersen *et al.* (2017), a simplified Real-Time Strategy game. We present deep maze, a flexible environment with a wide range of challenges suited for reinforcement learning research. The deep line wars environments feature a continuous environment where complex strategies must be planned. The deep maze environment provides simpler rules and a non-continuous action and state-space that in-comparison is far simpler then deep line wars.

5.1 | The deep maze environment

The deep maze is a flexible learning environment for controlled research in exploration, planning and memory for reinforcement learning algorithms. Maze solving is a well-known problem, and is used heavily throughout the RL literature Sutton *et al.* (1999), but is often limited to small

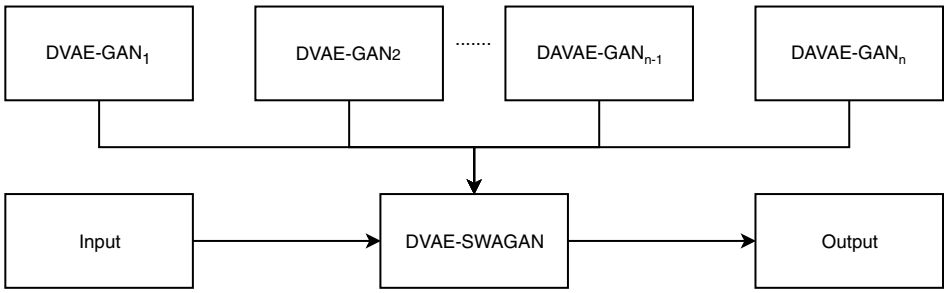


FIGURE 4 By using ensembles of DVAE-GAN models, the training is significantly more stable and prediction across sparse states yield better results for sequential input

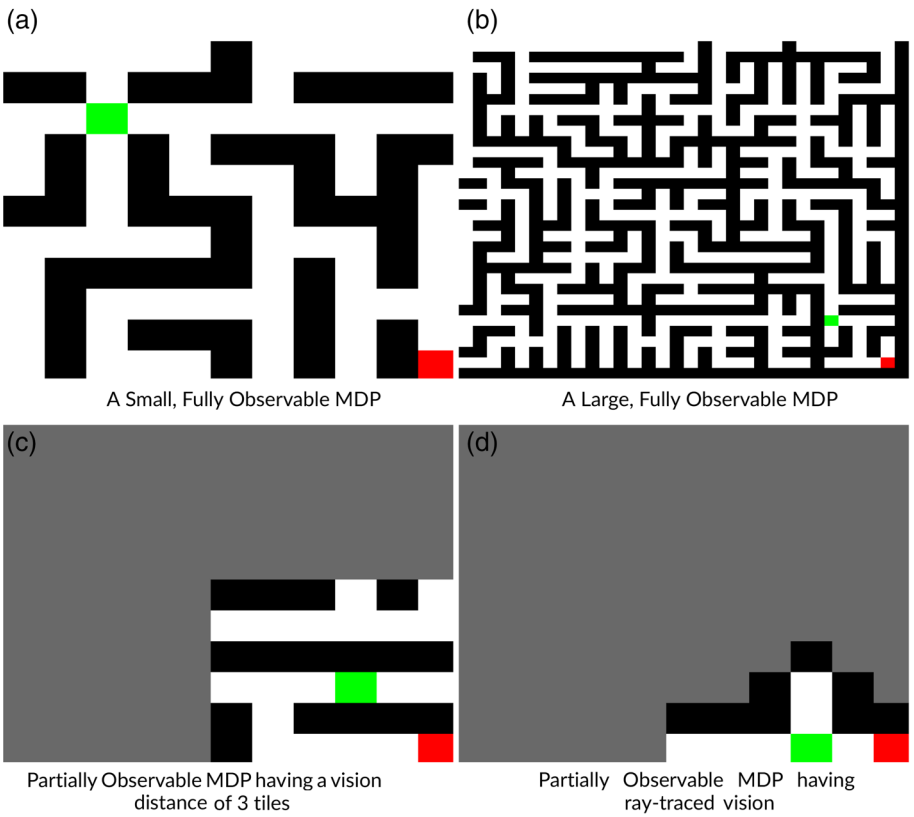


FIGURE 5 Overview of four distinct MDP scenarios using deep maze. (a) A small, fully observable MDP. (b) A large, fully observable MDP. (c) Partially observable MDP having a vision distance of three tiles. (d) Partially observable MDP having ray-traced vision

and fully observable scenarios. The deep maze environment extends the maze problem to over 540 unique scenarios including Partially Observable Markov Decision Processes (POMDP). Figure 5 illustrates a small subset of the available environments in the deep maze, ranging from small-scale MDP's to large-scale POMDP's. The deep maze further features custom game mechanics such as relocated exits and dynamically changing mazes. RL agents depend on sensory input to evaluate and predict the best action at the current timestep. Preprocessing of data is essential so that agents can extract features from the input data. For this reason, deep maze has built-in state representation for imaging and raw state matrices. The game engine is modularized and has an SDK that enables the development of third-party scenarios. This extends the capabilities of deep maze to support nearly all possible scenario combination in the realm of maze solving.[1]

The deep maze learning environment presents the following scenarios. (a) Normal, (b) POMDP, (c) Limited POMDP and (d) Timed Limited POMDP. The first mode exposes a seed-based randomly generated maze where the state-space an MDP. The second mode narrows the state-space observation to a configurable area around the player. In addition to radius-based vision, the POMDP mode also features ray-tracing vision that better mimic the sight of a physical agent. The third and fourth mode is intended for memory research where the agent must find the goal in a limited number of time-steps. In addition to this, the agent is presented with the solution but fades after a few initial time steps. The objective is for the agent to remember the solution to find the goal. All scenario setups have a variable map-size ranging between 2×2 and 56×56 tiles.

FIGURE 6 The graphical user interface of the deep line wars environment

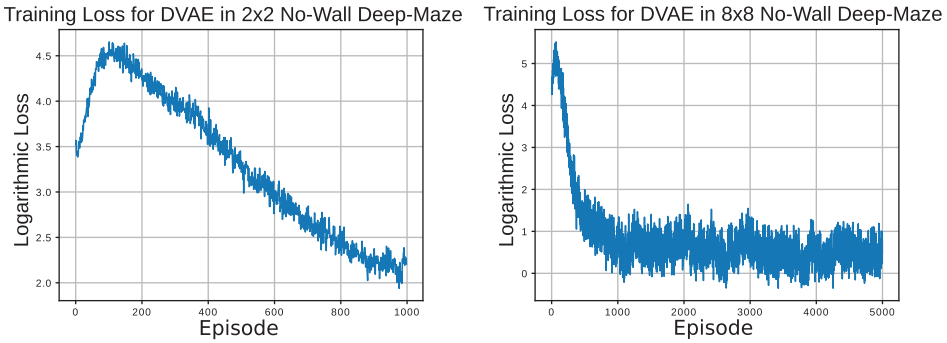


FIGURE 7 The training loss for DVAE in the 2×2 No-Wall and 8×8 deep maze scenario. The experiment is run for a total of 1000 (5000 for 8×8) epochs. The algorithm only trains on 50% of the state-space to the model for the 2×2 environment while the whole state-space is trainable in the 8×8 environment

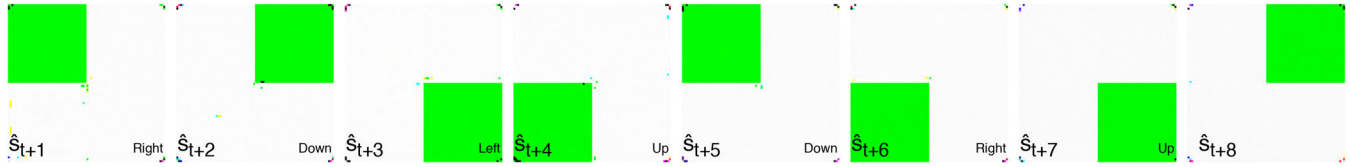


FIGURE 8 For the 2×2 scenario, only 50% of the environment is explored, leaving artefacts on states where the model is uncertain of the transition function. In more extensive examples, the player disappears, teleports or gets stuck in unexplored areas

5.2 | The deep line wars environment

The deep line wars environment was originally introduced in Andersen *et al.* (2017). Deep line wars is a real-time strategy environment that makes an extensive state-space reduction to enable swift research in reinforcement learning for RTS games.

The game objective of deep line wars is to invade the enemy player with mercenary units until all health points are depleted, see Figure 6. For every friendly unit that enters the far edge of the enemy base, the enemy health pool is reduced by one. When a player purchases a mercenary unit, it spawns at a random location inside the edge area of the buyers base. Mercenary units automatically move towards the enemy base. To protect the base, players can construct towers that shoot projectiles at the opponent's mercenaries. When a mercenary dies, a fair percentage of its gold value is awarded to the opponent. When a player sends a unit, the income is increased by a portion of the units gold value. As a part of the income system, players gain gold at fixed intervals.

6 | EXPERIMENTS

We centre our experiments around the DVAE, DVAE-GAN, DVAE-SWA and DVAE-SWAGAN. The goal of the proposed extensions is to improve the model stability so that the DVAE algorithm can produce better quality output for continuous and sparse state-spaces. In this section, we show results of model-based reinforcement learning using DVAE in the deep-maze and deep-line wars environment. We show the performance of encoding raw pixel input to a compact representation and to decode this representation to probable future states.

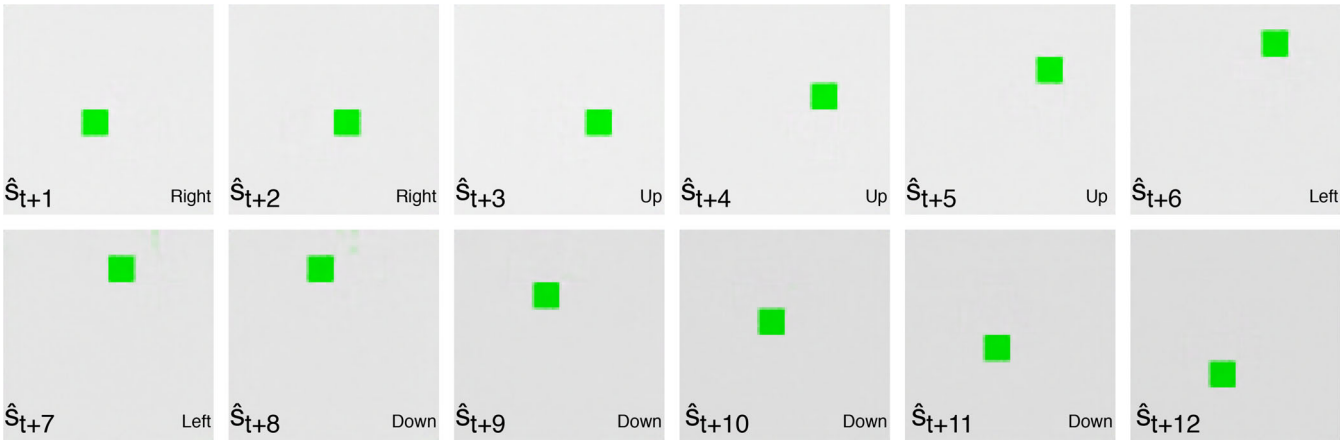


FIGURE 9 Results of 8 × 8 Deep Maze modelling using the DVAE algorithm. To simplify the environment, no reward signal is received per iteration. The left caption describes current state, s_t, while the right caption is the action performed to compute, $s_{t+1} = \mathcal{T}(s_t, a_t)$

TABLE 3 Results of the deep maze 11 × 11 and 21 × 21 environment, comparing DQN Mnih *et al.* (2015), TRPO Schulman *et al.* (2015) and PPO Schulman *et al.* (2017)

Algorithm	11 × 11	Average performance (%)	Converged epoch	21 × 21	Average performance (%)	Converged epoch
DQN-$\hat{\mathcal{D}}$		94.56	9314		64.36	N/A
TRPO-$\hat{\mathcal{D}}$		96.32	5320		78.91	7401
PPO-$\hat{\mathcal{D}}$		98.71	3151		89.33	7195
DQN-$\mathcal{D}$		98.26	4314		84.63	8241
TRPO-$\mathcal{D}$		99.32	3320		92.11	4120
PPO-$\mathcal{D}$		99.35	2453		96.41	2904

Note: The optimal path yields performance of 100% while no solution yields 0%. Each of the algorithms ran 10000 epochs for both map-sizes. Converged Epoch represents at which epoch the algorithm converged during training.

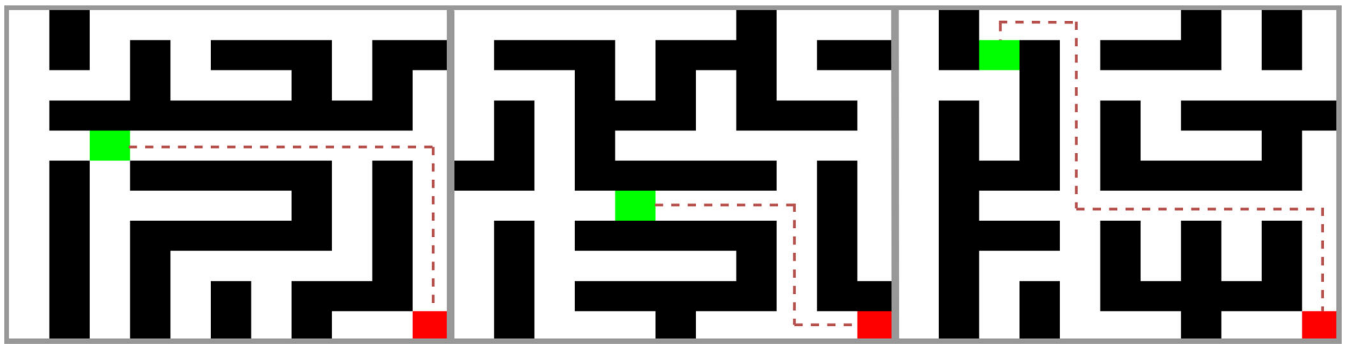


FIGURE 10 A typical deep maze of size 11 × 11. The lower-right square indicates the goal state, the dotted-line is a retrace of the predicted optimal path for that maze, while the final square represents the player's current position in the state-space. The controller agent is DQN, TRPO and PPO (from left to right)

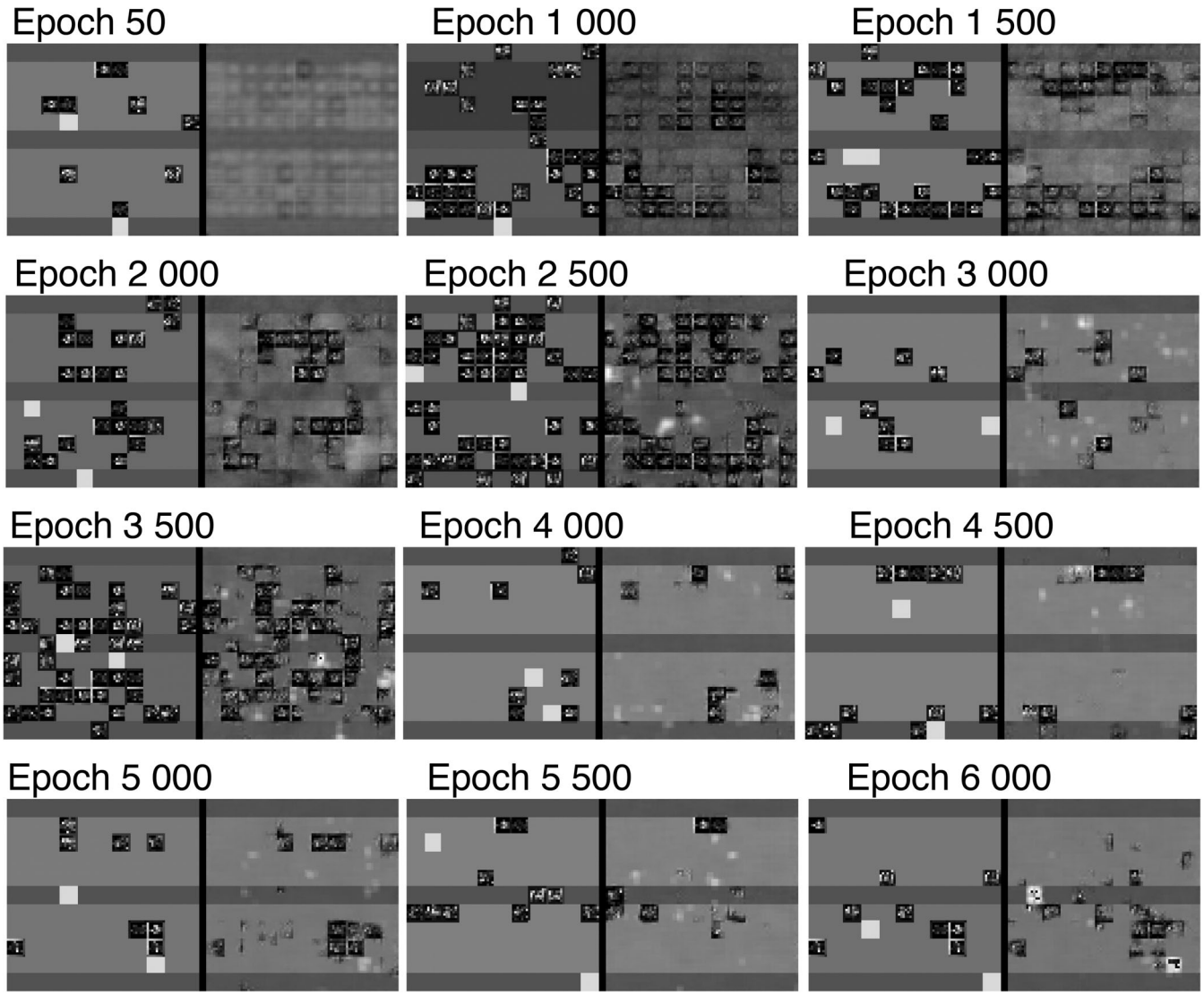


FIGURE 11 The DVAE algorithm applied to the deep line wars environment. Each epoch illustrates the quality of generated states in the game, where the left image is real state s and the right image is the generated state $\hat{s}$

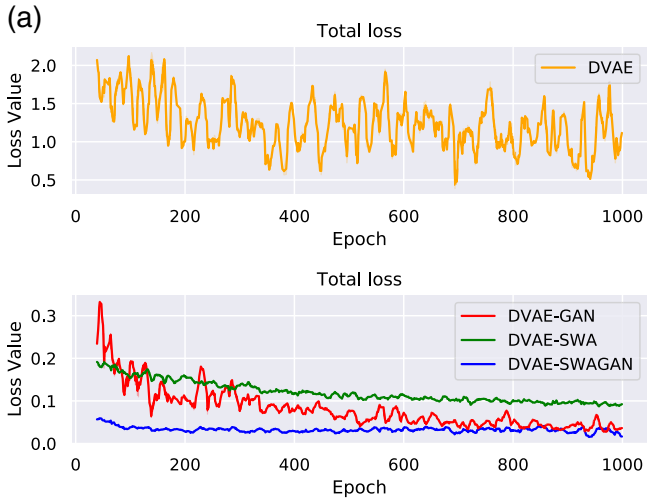
6.1 | The dreaming variational autoencoder

6.1.1 | Deep maze

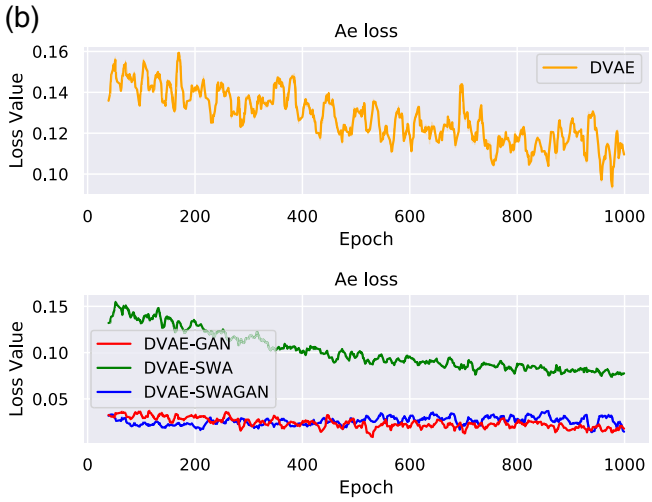
For the deep maze environment, the algorithm must be able to generalize over many similar states to model a large state-space. DVAE aims to learn the transition function $\mathcal{T}(s_t, a_t)$, bringing the state from s_t to s_{t+1}. We use the deep maze environment because it provides simple rules, with a controllable state-space complexity. Also, we can omit the importance of reward for some scenarios.

We trained the DVAE model on two *No-Wall deep maze* scenarios of size 2×2 and 8×8. For the encoder and decoder, we used the same convolution architecture as proposed by Pu *et al.* (2016) and trained for 5000 epochs[2] in the 8×8 scenario and 1000 epochs for 2×2, respectively. The reasoning behind different epoch lengths is because we expect simpler environments to converge faster. For the encoding of actions and states, we concatenate the flattened state-space and action-space, having a fully connected layer with ELU activation before calculating the latent-space. We used the Adam optimizer Kingma and Ba (2015) with a learning-rate of 3e-05 to update the parameters. To calculate the loss we used the same loss function as proposed by Kingma and Welling (2013).

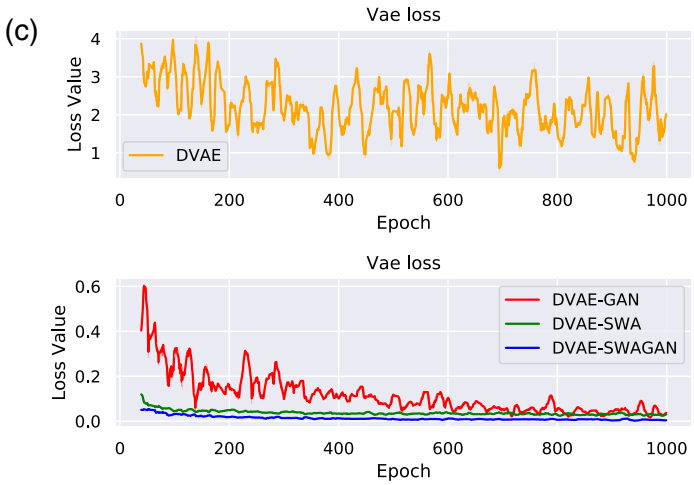
Figure 7 illustrates the loss during the training phase for the DVAE algorithm in the No-Wall Deep Maze scenario. In the 2×2 scenario, DVAE is trained on only 50% of the state space, which results in noticeable graphics artefacts in the prediction of future states, seen in Figure 8. In the 8×8 environment, the algorithm is allowed to train on all possible states, and we observe in Figure 9 that there is significantly better image quality throughout the sampling process.



Total Loss for the tested algorithms.



Autoencoder loss (L2) for the algorithms.



Variational Autoencoder Loss (KL) for the tested algorithms.

FIGURE 12 Training of DVAE compared to the DVAE-SWA, DVAE-GAN and DVAE-SWAGAN extension. (a) Total Loss for the tested algorithms. (b) Autoencoder loss (L2) for the algorithms. (c) Variational Autoencoder Loss (KL) for the tested algorithms

6.2 | Training RL-agents using $\hat{\mathcal{D}}$ replay-buffer

The goal of this experiment is to observe the performance of RL-agents using the generated experience-replay $\hat{\mathcal{D}}$ from Algorithm 1 in Deep Maze environments of size 11×11 and 21×21. In Table 3, we compare the performance of DQN Mnih *et al.* (2013), TRPO Schulman, Levine, Abbeel, Jordan and Moritz (2015) and PPO Schulman, Wolski, Dhariwal, Radford and Klimov (2017) using the DVAE generated $\hat{\mathcal{D}}$ to tune the parameters. Because TRPO and PPO are on-policy algorithms, the generated states must be generated on-the-fly so that the algorithm remains on-policy.

Figure 10 illustrates three maze variations of size 11×11, where the agent has learned the optimal path. We see that the best performing algorithm, PPO Schulman *et al.* (2017) beats DQN and TRPO using either $\hat{\mathcal{D}}$ or $\mathcal{D}$. The DQN-$\hat{\mathcal{D}}$ agent did not converge in the 21×21 environment, but it is likely that value-based algorithms could struggle to map inaccurate states with graphical artefacts generated from the DVAE algorithm. These artefacts significantly increase the state-space significantly, but empirical data suggest that on-policy algorithms perform better on noisy state-spaces.

6.3 | Deep line wars

The DVAE algorithm works well in more complex environments, such as the deep line wars game environment Andersen *et al.* (2017). Here, we expand the DVAE algorithm with LSTM to improve the capability of generating time-bound data, such as animations seen in Figure 1.

Figure 11 illustrates the state quality during training of DVAE in a total of 6000 epochs. Both players draw actions from a Gaussian distributed policy. The algorithm understands that the player units can be located in any tiles after only 50 epochs, and at 1000 epochs we observe the algorithm makes significantly better predictions of the probability of unit locations (i.e. some units show more densely in the output state). At the end of the training, the DVAE algorithm is to some degree capable of determining both towers, and unit locations at any given time-step during the game epoch.

6.4 | Extending the dreaming variational autoencoder

The goal of DVAE-SWA, DVAE-GAN and DVAE-SWAGAN is to perform better in continuous state-spaces, such as the deep line wars environment. The experiments were performed using a map size of 11×11 sampling actions from a PPO policy.

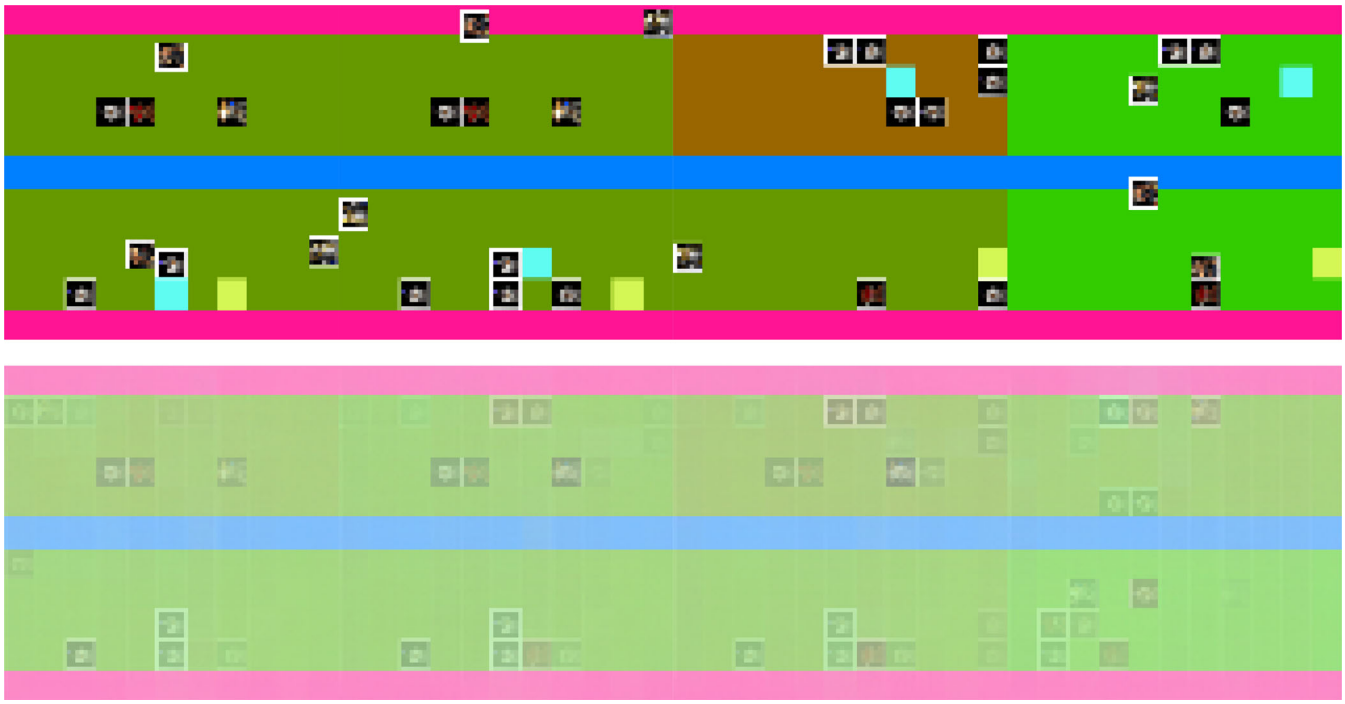


FIGURE 13 The first row represents the ground truth future state, while the second row is the predicted future state. The DVAE-SWAGAN algorithm sampled states after training for 1000 epochs, see Figure 12. Notice that the quality is notably better compared to Figure 11. We found that states were generalized too much producing some inaccurate predictions. Despite this inaccuracy, the RL agents learned a policy capable of beating random agents. We believe this is because similar states often represent similar value functions

Figure 12 shows the training loss of the algorithms DVAE, DVAE-SWA, DVAE-GAN and DVAE-SWAGAN for 1000 epochs (x-axis), where the y-axis describes the loss value. The new architectures perform significantly better than DVAE across all loss components of the architecture. The consequence of lower loss is better image quality (autoencoder loss), and better transitions (variational loss). For these experiments, we tried to model the deep line wars environment using RGB input. Figure 13 illustrates the resulting images for each of the algorithms. Here, we see that DVAE-SWAGAN perform significantly best, in terms of quality and accuracy.

7 | CONCLUSION AND FUTURE STUDY

This article introduces *The Dreaming Variational Autoencoder* along with its extensions DVAE-SWA, DVAE-GAN and DVAE-SWAGAN as a neural network-based generative modelling architecture to enable exploration in environments with sparse reward.

The DVAE algorithm successfully generates authentic world models in non-continuous state-spaces where the dynamics of the environment is simple. It works well for small environments but is limited when the state-sequence become too large. The algorithm performs marginally better when using LSTM for sequence prediction, but is a significant performance drop due to the increased model complexity. For most environments, such as deep line wars and deep maze, it is sufficient to run DVAE using only fully connected nodes.

The DVAE-SWAGAN improves the original model significantly and enables the algorithm to imitate environment models with continuous state-space better. DVAE-SWAGAN performs better in all environments including deep line wars and most environments found in the GYM reinforcement learning environment.

There are, however, several fundamental issues that limit DVAE, and DVAE-SWAGAN from fully modelling environments. In some situations, exploration may be a costly act that makes it impossible to explore all parts of the environment in its entirety. The algorithms cannot accurately predict the outcome of unexplored areas of the state-space, making the prediction blurry or incorrect. To combat this, the model should be improved further to include some sense of logic, and understanding of the environment dynamics. In current state-of-the-art, this frequently introduced as domain knowledge that is manually crafted by the programmer, but the hope is that future research will find a method for self-supervised domain knowledge modelling.

Reinforcement learning has many unresolved problems, and the hope is that the deep maze and the deep line wars learning environment can be a useful tool for future research. For future study, we plan to introduce an inverse reinforcement learning component to learn the reward function $\hat{\mathcal{R}}$. We also plan to explore non-parametric variants. DVAE and environment modelling is an ongoing research question, and the goal is that reinforcement learning algorithms could utilize this form of *dreaming* to make the algorithm far more sample efficient.

ORCID

Per-Arne Andersen https://orcid.org/0000-0002-7742-4907
Morten Goodwin https://orcid.org/0000-0001-6331-702X
Ole-Christoffer Granmo https://orcid.org/0000-0002-7287-030X

ENDNOTES

[1] The deep maze is open-source and publicly available at https://github.com/CAIR/deep-maze.

[2] We regard the term epoch as an episode, which is frequently in literature.

REFERENCES

Andersen, P.-A., Goodwin, M., & Granmo, O.-C. (2017). Towards a deep reinforcement learning approach for tower line wars. In M. Bramer & M. Petridis (Eds.), *Lecture notes in computer science (including subseries lecture notes in artificial intelligence and lecture notes in bioinformatics)* (Vol. 10630 LNAI, pp. 101–114) NY, USA: Springer, Cham. https://doi.org/10.1007/978-3-319-71078-5_8

Andersen, P.-A., Goodwin, M., & Granmo, O.-C. (2018, Dec). The dreaming variational autoencoder for reinforcement learning environments. In Max Bramer & M. Petridis (Eds.), *Artificial intelligence* (XXXV ed 11311, pp. 143–155). Springer, Cham. Retrieved from http://link.springer.com/10.1007/978-3-030-04191-5_11 doi: https://doi.org/10.1007/978-3-030-04191-5_11

Arulkumaran, K., Cully, A., & Togelius, J. (2019). *AlphaStar: An evolutionary computation perspective* (Tech. Rep.). Retrieved from https://deepmind.com/blog/alphastar-mastering-real-time-strategy-game-starcraft-ii/

Arulkumaran, K., Deisenroth, M. P., Brundage, M., & Bharath, A. A. (2017). Deep reinforcement learning: A brief survey. *IEEE Signal Processing Magazine*, *34* (6), 26–38. Retrieved from https://ieeexplore.ieee.org/document/8103164/. doi: https://doi.org/10.1109/MSP.2017.2743240

Azar, M. G., Piot, B., Pires, B. A., Grill, J.-B., Altché, F., & Munos, R. (2019, feb). *World discovery models. arxiv preprint arXiv:1902.07685*. Retrieved from http://arxiv.org/abs/1902.07685

Bangaru, S. P., Suhas, J., & Ravindran, B. (2016, nov). *Exploration for multi-task reinforcement learning with deep generative models. arxiv preprint arXiv:1611.09894*. Retrieved from http://arxiv.org/abs/1611.09894

Blundell, C., Uria, B., Pritzel, A., Li, Y., Ruderman, A., Leibo, J. Z., … Hassabis, D. (2016, jun). *Model-free episodic control. arxiv preprint arXiv:1606.04460*. Retrieved from http://arxiv.org/abs/1606.04460

Buesing, L., Weber, T., Racaniere, S., Eslami, S. M. A., Rezende, D., Reichert, D. P., … Wierstra, D. (2018, feb). Learning and querying fast generative models for reinforcement learning. *Arxiv Preprint arXiv:1802.03006*. Retrieved from http://arxiv.org/abs/1802.03006

Chua, K., Calandra, R., McAllister, R., & Levine, S. (2018, may). Deep reinforcement learning in a handful of trials using probabilistic dynamics models. *Advances in Neural Information Processing Systems 31*, 4759–4770. Retrieved from http://arxiv.org/abs/1805.12114

Ha, D., & Schmidhuber, J. (2018, sep). Recurrent world models facilitate policy evolution. *Advances in Neural Information Processing Systems 31*, 2455–2467. Retrieved from http://arxiv.org/abs/1809.01999

Hafner, D., Lillicrap, T., Fischer, I., Villegas, R., Ha, D., Lee, H., & Davidson, J. (2018, nov). Learning latent dynamics for planning from pixels. in *Proceedings of the 36th International Conference on Machine Learning*. Retrieved from http://arxiv.org/abs/1811.04551

Higgins, I., Matthey, L., Pal, A., Burgess, C., Glorot, X., Botvinick, M., … Lerchner, A. (2016, nov). beta-VAE: Learning basic visual concepts with a constrained variational framework. in *International Conference on Learning Representations*. Retrieved from https://openreview.net/forum?id=Sy2fzU9gl

Higgins, I., Pal, A., Rusu, A., Matthey, L., Burgess, C., Pritzel, A., … Lerchner, A. (2017). DARLA: Improving zero-shot transfer in reinforcement learning. In D. Precup & Y. W. Teh (Eds.), *in Proceedings of the 34th International Conference on Machine Learning* (Vol. 70, pp. 1480–1490). International Convention Centre, Sydney, Australia: PMLR. Retrieved from http://proceedings.mlr.press/v70/higgins17a.html

Izmailov, P., Podoprikhin, D., Garipov, T., Vetrov, D., & Wilson, A. G. (2018, mar). Averaging weights leads to wider optima and better generalization. Retrieved from http://arxiv.org/abs/1803.05407

Kaelbling, L. P., Littman, M. L., & Moore, A. W. (1996, apr). Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, *4*, 237–285. Retrieved from http://arxiv.org/abs/cs/9605103 doi: 10.1.1.68.466

Kingma, D. P., & Ba, J. L. (2015). Adam: A method for stochastic optimization. in *Proceedings*, *International Conference on Learning Representations* 2015. doi: https://doi.org/10.1145/1830483.1830503

Kingma, D. P., & Welling, M. (2013, dec). Auto-encoding variational Bayes. *arxiv preprint arXiv:1312.6114*. Retrieved from http://arxiv.org/abs/1312.6114 doi: https://doi.org/10.1051/0004-6361/201527329

Li, Y. (2017, jan). Deep reinforcement learning: An overview. *Arxiv preprint arXiv:1701.07274*. Retrieved from http://arxiv.org/abs/1701.07274

Liang, X., Wang, Q., Feng, Y., Liu, Z., & Huang, J. (2018, dec). VMAV-C: A deep attention-based reinforcement learning algorithm for model-based control. *arxiv preprint arXiv:1812.09968*. Retrieved from http://arxiv.org/abs/1812.09968

Makhzani, A., Shlens, J., Jaitly, N., Goodfellow, I., & Frey, B. (2015, nov). Adversarial autoencoders. Retrieved from http://arxiv.org/abs/1511.05644

Mnih, V., Badia, A. P., Mirza, M., Graves, A., Lillicrap, T., Harley, T., … Kavukcuoglu, K. (2016). Asynchronous methods for deep reinforcement learning. In M. F. Balcan & K. Q. Weinberger (Eds.), *Proceedings of the 33rd International Conference on Machine Learning* (Vol. 48, pp. 1928–1937). New York, USA: PMLR. Retrieved from http://proceedings.mlr.press/v48/mniha16.html

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., & Riedmiller, M. (2013, dec). Playing atari with deep reinforcement learning. *Neural Information Processing Systems*. Retrieved from http://arxiv.org/abs/1312.5602

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves A., Riedmiller M., Fidjeland A. K., Ostrovski G., Petersen S., Beattie C., Sadik A., Antonoglou I., King H., Kumaran D., Wierstra D., Legg S. Hassabis, D. (2015, feb). Human-level control through deep reinforcement learning. *Nature*, *518*(7540), 529–533. Retrieved from https://doi.org/10.1038/nature14236

Mousavi, S. S., Schukat, M., & Howley, E. (2018). Deep reinforcement learning: An overview. In Y. Bi, S. Kapoor, & R. Bhatia (Eds.), *Proceedings of Sai Intelligent Systems Conference (Intellisys) 2016* (pp. 426–440). Cham: Springer International Publishing. Retrieved from http://link.springer.com/10.1007/978-3-319-56991-8_32

Pu, Y., Gan, Z., Henao, R., Yuan, X., Li, C., Stevens, A., & Carin, L. (2016). Variational autoencoder for deep learning of images, labels and captions. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, & R. Garnett (Eds.), *Advances in neural information processing systems* (pp. 2352–2360). NY, USA: Curran Associates, Inc. Retrieved from http://papers.nips.cc/paper/6528-variational-autoencoder-for-deep-learning-of-images-labels-and-captions.pdf

Schulman, J., Levine, S., Abbeel, P., Jordan, M., & Moritz, P. (2015). Trust region policy optimization. In F. Bach & D. Blei (Eds.), *Proceedings of the 32nd International Conference on Machine Learning* (Vol. 37, pp. 1889–1897). Lille, France: PMLR. Retrieved from http://proceedings.mlr.press/v37/schulman15.html

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017, jul). Proximal policy optimization algorithms. *arxiv preprint arXiv:1707.06347*. Retrieved from http://arxiv.org/abs/1707.06347

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., … Hassabis, D. (2017). Mastering the game of go without human knowledge. *Nature*, *550*, 354–359. https://doi.org/10.1038/nature24270

Smith, L. N. (2015, jun). Cyclical learning rates for training neural networks. Retrieved from http://arxiv.org/abs/1506.01186

Sutton, R. S., Precup, D., & Singh, S. (1999). *Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning* (Vol. 112; Tech. Rep.).

Xiao, T., & Kesineni, G. (2016). *Generative adversarial networks for model based reinforcement learning with tree search (Tech. Rep.)*. Berkeley: University of California. Retrieved from http://tedxiao.me/pdf/gans_drl.pdf

Zhang, C., Patras, P., & Haddadi, H. (2018, mar). Deep learning in mobile and wireless networking: A survey. *IEEE Communications Surveys and Tutorials*, *21*, 2224–2287. Retrieved from http://arxiv.org/abs/1803.04311

AUTHOR BIOGRAPHIES

Per-Arne Andersen is a Ph.D. candidate in Artificial Intelligence at the University of Agder, Norway where he also completed his B.Sc. and M.Sc. in 2015 and 2017, respectively. He has expertise in several domains in computer science, including security, software development and machine learning. His primary field of research is reinforcement learning in environments with incomplete information, including real-time games and industry-near systems. Towards the future, Per-Arne hopes to have an active role in the scientific community for reinforcement learning and to contribute towards the understanding of artificial general intelligence.

Morten Goodwin received the B.Sc. and M.Sc. degrees from University of Agder, Norway, in 2003 and 2005, respectively, and the Ph.D. degree from Aalborg University Department of Computer Science, Denmark, in 2011, on applying machine learning algorithms on eGovernment indicators which are difficult to measure automatically. He is an Associate Professor with the Department of ICT, University of Agder, deputy director for Centre for Artificial Intelligence Research, coordinator for the International Master's Programme in Artificial Intelligence, a public speaker and an active researcher. His main interests include swarm intelligence, deep learning and adaptive learning in the fields of medicine, games and chatbots. He has more than 70 peer reviews scientific publications, and has supervised more than 110 student projects including Master's and Ph.D. theses.

Prof. Ole-Christoffer Granmo is director and founder of the Centre for Artificial Intelligence Research (CAIR) at the University of Agder, Norway. He obtained his master's degree in 1999 and the Ph.D. degree in 2004, both from the University of Oslo, Norway. Granmo develops theory and algorithms for systems that explore, experiment and learn in complex real-world environments. His research interests include artificial intelligence, machine learning, learning automata, bandit algorithms, deep reinforcement learning, Bayesian reasoning and computational linguistics. Within these areas of research, Dr. Granmo has written more than 125 refereed journal and conference publications. He is co-founder of the Norwegian Artificial Intelligence Consortium (NORA). Apart from his academic endeavours, Granmo is also co-founder of the company Anzyz Technologies AS.

How to cite this article: Andersen P-A, Goodwin M, Granmo O-C. Increasing sample efficiency in deep reinforcement learning using generative environment modelling. *Expert Systems*. 2020;e12537. https://doi.org/10.1111/exsy.12537