Towards Safe and Sustainable Reinforcement Learning for Real-Time Strategy Games

Per-Arne Andersen*, Morten Goodwin, Ole-Christoffer Granmo

Centre for Artificial Intelligence Research, Department of ICT, University of Agder, Jon Lilletuns vei 9, 4879 Grimstad, Norway

Abstract

The combination of Deep Neural Networks and Reinforcement Learning, namely Deep Reinforcement Learning (DRL), shows continued success in solving complex problems across many areas such as medicine, industry, and game playing. Perhaps most notably, DRL has shown outstanding performance in advanced Real-Time Strategy (RTS) games such as StarCraft II and Dota 2, arguably the most challenging games used in RL literature to date. RTS games have highly uncertain environments with little observable information, making them perfect for evaluating the robustness and safety of RL algorithms.

There are, however, still significant limitations with DRL algorithms in the literature, such as an ever-increasing computational cost and little focus on safety-aware approaches. Most published algorithms are computationally expensive to train, making it near-impossible to retrain without significant resources. A consequence is an ever-increasing gap between state-of-the-art algorithms trained on supercomputers and algorithms trained on commodity hardware. Training these computationally intensive DRL algorithms impacts CO2 emission significantly, which is arguably not sustainable. A majority of RL algorithms in the literature are risk-neutral, are demanding to deploy in safety-critical systems, and can result in catastrophic failures. While there are some efforts to improve RL safety, few methods transfer well from theoretically verified scenarios to complex real-world autonomous systems validated experimentally.

This article presents a novel model-based DRL approach for tackling complex environments with the aim to reduce the need for failures during training. Specifically, our approach demonstrates successful learning while still considering robust safety awareness, minimizing risk, and reducing computational costs compared to model-free RL methods.

Our approach, the **S**afe **O**bservations **R**ewards **A**ctions **C**osts **L**earning **E**nsemble (S-ORACLE), is empirically verified in multiple complex and uncertain game enviroments: Deep RTS, ELF: MiniRTS, MicroRTS, Deep Warehouse, and StarCraft II, outperforming state-of-the-art model-free and model-based approaches.

Keywords: Reinforcement Learning, Markov Decision Processes, Neural Networks, State-space models, Model-based Reinforcement Learning, Risk-aware RL, Mission-critical safety

*Corresponding author
Email address: `per.andersen@uia.no` (Per-Arne Andersen)

1. Introduction

The desired property of artificial intelligence is its ability to solve complex real-world challenges. Reinforcement Learning (RL) is a field of study which concerns agents that ought to do sequential decision making in a dynamic system or environment. RL takes inspiration from nature and the art of learning through trial and **error** and seeks to learn optimal behavior policy adjusting through feedback after each consecutive action. The theoretical background of RL stems from Markov chains, with augmentations to form Markov decision processes (MDP) [1]. This mathematical framework describes the probability of transitioning from one state to another given a decision. However, because RL aims to maximize rewards long-term, the likelihood of entering catastrophic states increases dramatically. RL classifies into two categories of algorithms. [2]

• **Model-free RL (MFRL)** algorithms aim to find the behavior policy without a known transition probability distribution and the corresponding reward function but instead learn through balancing the exploration/exploitation dilemma under the assumption of an unknown MDP [2].

• **Model-based RL (MBRL)** aims to learn its policy through a known model of the environment. It consists of a state transition probability function and a reward function and is referred to as a dynamics model in the literature. A dynamics model is either known *a priori* or learned through estimations based on observations and interactions with the environment. In real-world and complex computational tasks, the complete state information is rarely available, and hence, the estimated dynamics models often inherit the uncertainty about missing information. Model-based RL is shown to have better sample efficiency and more flexibility for safe RL, which are appealing traits for mission-critical and sustainable applications. [3]

• **Safety** is a central problem to RL because algorithms deployed in the real world risk damaging equipment or humans during the learning process. This gives us clear motivation to address these shortcomings in the realm of **safe reinforcement learning** (safe RL) [4]. Safe RL aims to learn a policy that maximizes rewards while also maintaining safety. There are several directions in the literature towards safer RL, and we discuss this more closely in Section 3.

• **Sustainability** meets the needs of the present without compromising the ability of future generations to meet their own needs. Machine Learning has a notorious reputation for an ever-increasing computational cost, demonstrated well through reports of state-of-the-art results using millions of dollars in computing to achieve high accuracy [5]. The computation growth is not sustainable and produces a high amount of climate emission gasses during experiments, negatively impacting the environment. However, recent literature from Henderson et al. proposes a systematic method of reporting climate emission for experiments that aim to reduce the overall climate footprint of machine learning research. [6] To the best of our knowledge, this is the first article presenting safe reinforcement learning that systemically reports climate emission, detailed in Section 6.3.

- **Recent literature** shows that RL can perform at a superhuman level several complex problems [7] such as robotic control [8]; optimizing portfolio and performing algorithmic trading in stock markets [9]; database optimization [10]; planning and path optimization [11]; medical applications such as breast cancer classification and biological sciences [12, 13]; autonomous control in vehicles [14, 15]. Much of the recent success of applied RL stems from research into solving complex game environments. Recent literature shows that RL algorithms can learn purely from pixels, such as DQN in the Atari 2600 environment suite [16], mastering board games such as Chess and Go using a combination of tree-search and Deep RL [17, 18]. RL can likewise learn highly complex games such as Real-Time Strategy (RTS) games, a highly complex stochastic environment with near-infinite state spaces and action spaces [5]. While the mentioned studies perform well according to a reward signal, no approach exists to maximize **safety**, minimize **risk**, or address the unsustainable computational costs during training and inference, which real-world applications often demand.

1.1. Challenges

Although there is steady advancement in solving complex tasks using RL, there are many challenges left unchecked in RL algorithms for the broad adoption of industry, real-world, and safety-critical applications. These challenges include:

- safe learning without *a priori* knowledge [4],
- sample efficiency [2],
- high carbon emission footprint [6],
- exploration (trial and error)-exploitation dilemma [2],
- challenging to design reward structures [19],
- hyperparameter-sensitivity [20],
- and a clear gap between affordable RL and state-of-the-art [21]

It is well understood that RL requires millions of training samples to converge in complex tasks, and hence it naturally follows a high cost of computational power. The high computational power increases the demand for hardware which directly impacts the climate footprint negatively [6]. Furthermore, it is seldom that an RL algorithm functions well without extensive hyperparameter tuning and careful reward design; hence requires several training sessions to find a composition that yields an excellent behavioral policy. Combining these problems makes it difficult to reproduce algorithms and makes RL less suitable for applied intelligence in the industry. The consequence is that most RL research concerns risk-neutral algorithms with the primary goal of performing well in simulated environments, with little regard for applicability in safety-critical systems. Lastly, there is a significant gap in the literature where prior state-of-the-art solved simple RTS game setups

Table 1: RTS games share many characteristics with Real-World applications. The table draws parallels between central challenges in Real-world and RTS games.

Challenge	Real World	RTS Game
Reward Sparsity	Taking actions towards less climate emissions are rewarded several years later. [6]	Immediate actions are rewarded when a game ends and not after each consecutive action [19]
Partial Observability	Autonomous driving having an unknown number of participants in the traffic with the uncertainty of which actions other cars will make. [14]	Parts of the game are hidden in "Fog of War". The opponent makes changes to the environment without the agent knowing. [22]
High Dimensionality	Data from patients with thousands of dimensions (features) or high-resolution images from cameras [13]	RTS games have large maps (e.g., 256x256) with the ability to have 500 units with 100 different actions at each timestep. [5]
Planning over long time horizons	Autonomous driving requires planning from the start point to the destination point [12]	To succeed in RTS games, the player must plan ahead hundreds of time steps to beat the opponent.[5]
Learning Safely	Learning to drive a car happens under the supervision of an experienced (arguably expert) driver, and no accident must occur during the learning phase or after learning to drive the car.	Many RTS games have environmental damage that regresses the agent's progress. The agent must learn to avoid this damage, preferably without damage ever occurring. [20]

such as MicroRTS [22], and DeepRTS [23] wherein contrast, the latest state-of-the-art beats the world champion in Dota 2 [24] and achieves Grandmaster rank in Starcraft II [5]. We believe that the research in RTS games is far from solved and by no means mastered, considering that recent state-of-the-art does not transfer well to real-world applications and is not safe, affordable, or sustainable in long-term operations.

1.2. Motivation

Solving a complex game environment is an exciting feat because it illustrates that RL gradually matures towards the ultimate challenge of solving advanced and complex real-world problems in a safe manner[21]. RL is still in its infancy, and while many fundamental problems are solved, there is still room for improvement in sample efficiency, safety, and generalization [7]. In the interim, RTS games are an appealing platform to benchmark algorithms because they share similar characteristics to real-world problems [25]. Table 1 demonstrate common characteristics between RTS games and real-world dynamics. It falls natural to use RTS games as a platform for researching task solvability to find techniques applicable in safety-critical real-world applications [26]. The hope is that RL can play a substantial role in solving safety-critical real-world applications through learning problems virtually. These problems range from minor optimizations such as vehicle routing to major problems such as driving autonomously in traffic and optimizing the climate emissions of the engine exhaust system [27].

1.3. Contributions

The contribution of this article is composed of four highlights:

1. This work presents the **Safe O**bservations **R**ewards **A**ctions **C**osts **L**earning **E**nsemble (S-ORACLE), a novel model-based RL approach that aims towards **safer**, more sustainable, and more efficient performance. The algorithm uses an ensemble of originally risk-neutral model-free approaches to make decisions motivated using risk-averse reward signals and is trained purely on a model learned on a fraction of samples compared to only using model-free training.[1]
2. We perform a thorough performance evaluation of the proposed algorithm in RTS problems against well-known model-free approaches such as DQN, PPO, A2C, and RAINBOW in six environments; Deep RTS, Deep Line Wars, ELF: MiniRTS, StarCraft II, Deep Warehouse, and MicroRTS. The work presents baseline results of obtained performance in the tested environments and extensively evaluates the carbon emissions budget for used algorithms in the tested environment.
3. Safety of the proposed method is evaluated in the Deep RTS Lava environment and Deep Warehouse environment, an industry-like grid-warehouse simulator. The work demonstrates that the proposed algorithm outperforms risk-neutral RL algorithms while also being significantly more sample efficient.
4. Lastly, safety is discussed in context to RL, what the limits are and possible approaches towards truly generalizable safe algorithms, and concludes the findings of this work.

1.4. Outline

This article is organized in the following manner. Section 2 thoroughly details the theoretical background of S-ORACLE. It details Markov decision processes, model-free RL, model-based RL, and risk-aware RL techniques. Section 3 outlines a selection of fundamental research in the field of safe RL and explains the concepts thoroughly. Section 4 presents the RTS and safety-critical environments used in the evaluation of S-ORACLE and summarizes the configurations utilized during experiments. Section 5 presents the S-ORACLE algorithm and details the algorithm thoroughly seen from the perspective of safety and performance. Section 6 details the empirical evaluation of S-ORACLE in comparison to other state-of-the-art

[1]The source code is located at `https://github.com/s-oracle/s-oracle`

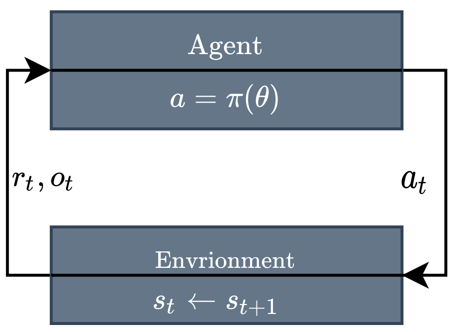


Figure 1: The agent-environment synergy in a Markov decision process [2]. The agent makes actions in the environment, which transitions the environment to the next state. The agent observes the new state with the corresponding reward signal.

algorithms. Specifically, the Section presents safety evaluations and performance evaluations. Lastly, we report and document the sustainability in terms of computational costs and climate emissions. Section 7 reflects on performance and safety and possibilities of safety-aware RL. Finally, Section 8 concludes the presented work and lays a path for future research.

2. Background

2.1. Markov Decision Processes

The essence of Reinforcement Learning is learning machine algorithms to make a sequence of decisions in a dynamic system. Markov decision processes (MDP's) are a mathematical framework that defines a class of stochastic sequential decision processes and are the fundamental building block of RL algorithms. In this article, we are concerned with games with a finite number of states and actions, and hence, we consider finite MDP's. The MDP framework is visualized in Figure 1 as a discrete-time sequential process of making decisions and then observe o_t with its corresponding reward r_t. The observation is either fully-observable such that $o_t = s_t$ or partially-observable such that $o_t \neq s_t$. [28]

Definition 1. *An MDP model is expressed as a tuple* $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$ *where* $\mathcal{S}$ *is the state space,* $\mathcal{A}$ *is the action space available to the agent at every time-step,* $\mathcal{P} : \mathcal{S} \times \mathcal{R} \times \mathcal{S} \times \mathcal{A} \rightarrow [0, 1]$ *is the transition function,* $\mathcal{R} : S \times \mathcal{A} \rightarrow \mathbb{R}$ *is the reward function, and* $\gamma \rightarrow [0, 1]$ *is the discount-factor [29].*

The transition function $\mathcal{P}$ with four-arguments and describes the probability of transitioning from a particular state s to s' with the corresponding reward r after taking action a. The transition function contains all information about the MDP,

$$\begin{aligned} \mathcal{P}(s', r|s, a) &= Pr\left[S_{t+1} = s', R_{t+1} | S_t = s, A_t = a\right] \\ &= \sum_{s' \in \mathcal{S}} \sum_{r \in \mathcal{R}} p(s', r|s, a) = 1, \end{aligned} \tag{1}$$

where $\mathcal{P}$ is defined for the next state s', $\forall s \in \mathcal{S}$, and $\forall a \in \mathcal{A}(s)$. From the four-argument transition function, we can derive a state-transition function $\mathcal{T} : \mathcal{S} \times \mathcal{S} \times \mathcal{A} \rightarrow [0, 1]$ and a reward function $\mathcal{R} : S \times \mathcal{A} \rightarrow \mathbb{R}$ for all state-action pairs. The state-transition function is defined,

$$\begin{aligned} \mathcal{T}(s'|s, a) &= Pr\left[S_t = s'|S_{t-1} = s, A_{t-1} = a\right] \\ &= \sum_{r \in \mathcal{R}} \mathcal{P}(s', r|s, a), \end{aligned} \tag{2}$$

where the probability of entering the next state s' is dependent on the current state s and the taken action a. The reward function is defined,

$$\begin{aligned} \mathcal{R}(s, a) &= \mathbb{E}\left[R_{t+1}|S_t = s, A_t = a\right] \\ &= \sum_{r \in \mathcal{R}} r \sum_{s' \in \mathcal{S}} \mathcal{P}(s', r|s, a), \end{aligned} \tag{3}$$

where $\mathcal{R}$ is the expected reward that the agent receives after making action a to transition to state s'. An environment has the Markov property if it is possible to predict the next state and expected next reward given only the current state and action. The Markov property exists for a MDP only if,

$$\begin{aligned} &Pr\left[S_{t+1} = s', R_{t+1} = r|s_t, a_t, \ldots s_0, a_0\right] \\ &= Pr\left[S_{t+1} = s', R_{t+1} = r|s_t, a_t\right]. \end{aligned} \tag{4}$$

For complex game environments or real-world applications, the Markov property is rarely present because full observations of the system dynamics are hidden ($o_t \neq s_t$), making it difficult to characterize the next state given the current observed state fully. [30].

2.2. Partially Observable MDP

In complex games such as Starcraft II and Dota 2, only partial information of the state is accessible and, therefore, the agent must use that partial observation or construct beliefs of what the true state may be. Partially Observable Markov Decision Processes (POMDP)'s are a generalization of MDPs that accounts for partial observability of the state and is therefore widely used in literature to formalize optimization problems in game environments. POMDP is defined as a tuple $\mathcal{M}_{\mathcal{POMDP}} = \langle \mathcal{S}, \mathcal{A}, \Omega, \mathcal{P}, \mathcal{R}, \mathcal{O}, \gamma \rangle$ of three sets and three functions. The definition is similar to regular MDP's but extends with a set of observations $\Omega = \{o_1, o_2, \ldots, o_n\}$ and the agent perception model $O : \mathcal{S} \times \mathcal{A} \rightarrow \Pi(\Omega)$ where $\Pi(\Omega)$ represent the probability distribution on Ω. [31]. Figure 2 illustrates a POMDP where the agent only can make actions a_t that are dependent on observations o_t. The problem with observations is that they may not capture the necessary information to succeed in the environment. For example, if we consider a game of pong where the observation is a pixel image, the agent cannot determine the direction or velocity of the ball. To alleviate these problems, the agent can make decisions based on a history of observations. However, keeping a history of prior observations is infeasible for games with near-infinite states and hence, is not a sufficient method for complex games.

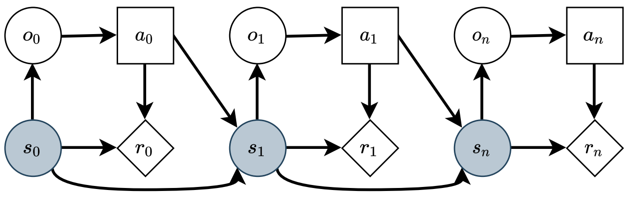


Figure 2: A POMDP system. o, a, s, r denotes observations, actions, states, and rewards, respectively. The agent can only observe $o_0 \dots o_n$ after making an action where the underlying state $s_t \dots s_n$ is hidden from the agent.

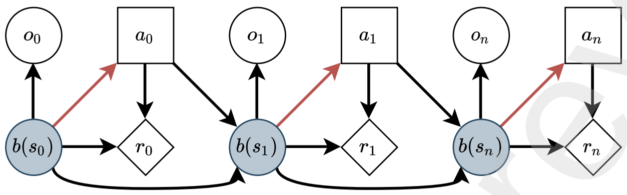


Figure 3: A Belief MDP. b, a, s, r denotes belief-state distributions, actions, states, and rewards, respectively. In contrast to POMDP's, the underlying state s is replaced with a belief state distribution. States sampled from the belief state distribution are used to make decisions in the MDP, which has similar traits to fully observable MDP's.

2.3. Belief MDP

Another approach to capture information from history is to encode observations into a belief state using belief MDPs. 155 A belief state b is a summarization of previous observations into a probability distribution over all states $s \in \mathcal{S}$, where $b(s) = Pr(s_t|o_{1\dots t})$, represents the probability that the environment is in state s [32]. Given that we have an initial belief state b_0, we can compute belief states,

$$\begin{aligned} b'(s') &= P(s'|o, a, b) \\ &\propto P(o|s', a, b)P(s'|a, b) \\ &\propto O(o|s', a)P(s'|a, b) \\ &\propto O(o|s', a) \sum_{s \in \mathcal{S}} P(s'|a, b, s)P(s|a, b) \\ &\propto \underbrace{O(o|s', a)}_{\text{Observation Model}} \sum_{s \in \mathcal{S}} \underbrace{\mathcal{T}(s'|s, a)}_{\text{State-Transition}} \underbrace{b(s)}_{\text{belief}}, \end{aligned} \tag{5}$$

which is a sufficient statistic for $b_t \equiv o_{1\dots t}$. The reward function $R(s, a)$ requires hidden-state information, but we can introduce belief states such that $R(b, a) = \sum_{s \in \mathcal{S}} b(s)R(s, a)$. The belief state distribution b can 160 memorize historical observations and we assume that this is enough information to represent the hidden-state s_t so that we can treat the POMDP similarly to fully-observable MDP's, seen in Figure 3. [31, 33]

2.4. Reinforcement Learning

Analogous to human intelligence, a behavior policy is considered the brain of the algorithm and expresses parameters for a function that aims to behave optimally in a given problem. This Section is dedicated to explain classical RL commonly uses tables or simple function approximators to parameterize the behavior policy. These methods are shown to work well for simple problems, but for games or real-world applications, both fall inadequate because of large state and action spaces. [2]

The ultimate goal of a reinforcement learning agent is to find the optimal policy π^*. A policy represents a mapping from a state observation to action probabilities $\pi(a|s)$. RL algorithms categorize into three learning types; value-based, policy-based, and a combination called actor-critic-based algorithms, where our work is based on value-based approaches. There are also variations of on-policy and off-policy where off-policy algorithms can learn using historical data. Lastly, there are model-based and model-free algorithms where model-based algorithms learn using a predicted model or a known environment model, in contrast to model-free algorithms that learn solely by trial and error in an unknown environment. An RL problem is commonly modeled as an MDP and usually have an update procedure as follows:

1. Read current observation s_t
2. Make a decision based on observation $\pi(a|s_t)$
3. Receive reward r_t
4. Update policy estimates with a learning algorithm. Go back to step 1.

The optimal policy π^* is the policy in policy-space that maximize the return,

$$\pi^* = \arg\max_{\pi \in \Pi} \mathbb{E}\left[G|\pi\right] \tag{6}$$

where the return G_t,

$$\begin{aligned} G_t &= R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} \\ &= \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}, \end{aligned} \tag{7}$$

is the cumulative discounted return. The discount factor $0 \leq \gamma \leq 1$ quantifies the importance between immediate rewards and distant rewards where $\gamma = 0$ considers only immediate rewards and $\gamma = 1$ weights immediate and distant rewards equally. [2]. Perhaps the most central Equation of RL is the Bellman Expectation Equations,

$$\begin{aligned} v_\pi(s) &= \mathbb{E}_\pi\left[G_t|S_t = s\right] \\ &= \mathbb{E}_\pi\left[R_{t+1} + \gamma v_\pi(S_{t+1})|S_t = s)\right] \end{aligned} \tag{8}$$

where $v_\pi(s)$ quantifies how good it is for the agent to be in state s_t following policy π henceforth [15]. The Bellman Equation, famously from dynamic programming, defines a recursive function that expresses the relationship between the value of a state and the successor state [34, 2]. However, the state-value function is not practical when quantifying how good actions are in state s_t because it summarizes all actions. The state-action value function $Q : S \times A \to \mathbb{R}$ quantifies how good action a_t being in state s_t. The state-action value function,

$$\begin{aligned} &Q_\pi(s,a) \\ &= \mathbb{E}_\pi\left[R_{t+1} + \gamma v(S_{t+1}) | S_t = s, A_t = a\right] \\ &= \mathbb{E}[R_{t+1} + \gamma \mathbb{E}_{a\sim\pi} Q(S_{t+1}, a) \mid S_t = s, A_t = a], \end{aligned} \tag{9}$$

is similar to the state-value function $Q_\pi(s,a)$ but decomposes the value-function into values for individual actions. [35]. Consequently, we know that the optimal policy π^* is found indirectly if $Q^*(s,a)$ or $V^*(s)$ is known through solving the Bellman Equation,[2] [7],

$$Q^*(s,a) = \mathbb{E}[R_{t+1} + \gamma \max_{a'} Q^*(S_{t+1}, a)]. \tag{10}$$

2.5. Model-Based RL

Model-based RL follows the standard MDP derivation and involves learning the transition-function $\mathcal{T}$ from observed data [37]. The goal is to find some parameters θ_m so that the estimated transition function $\hat{\mathcal{P}}(s', r|s, a; \theta_m) \cong \mathcal{P}(s', r|s, a)$. In this work, we learn to derive the reward-function $\hat{\mathcal{R}} : \mathcal{S} \times \mathcal{A}$ and state-transition function $\hat{\mathcal{T}} : \mathcal{S} \times \mathcal{A}$ because we aim to quantify the uncertainty of the reward function estimates for risk-sensitive RL, which is further detailed in Section 2.9. While it is common to incorporate decision-making into the model with algorithms such as cross-entropy methods (CEM), we have a model agnostic approach that allows model-free algorithms (e.g., Q-learning). Model-free approaches are more studied, hence has a significantly better performance than model-based approaches [5]. Contrary to model-free algorithms, model-based algorithms are much more sample efficient [38, 39] and in combination with a learned dynamics model, the aim is to get the best of two worlds: sample efficiency, performance, and risk-awareness incorporated together.

• **Dynamics Model.** The goal of an estimated dynamics model $\hat{\mathcal{T}} : \mathcal{S} \times \mathcal{A}$ is to learn parameters θ_m that best can reflect the behavior of the unknown dynamics $\mathcal{T}$ from Equation 2. The estimated predictive model is with this referred to as a dynamics model and is defined,

$$\hat{\mathcal{T}}(\hat{s}'|\hat{s}, a; \theta_m) \cong \mathcal{T}(s'|s, a) \tag{11}$$

where we assume that the estimated model is in some way captures information of the unknown MDP similar to HMM's [40]. There are many approaches to learning such models, and this article focuses on a combination of variational autoencoders (VAE) [41], state-space models [42], and recurrent neural networks [43].

[2]RL theory has significantly more ground to cover, but we have left out non-essential parts for this article. We recommend [36] for further reading.

• **Variational Autoencoder** is a generative autoencoder that uses neural networks to parametrize probability distributions used to define a probabilistic latent variable model efficiently. VAE's define a generative model and an inference network used to learn the parameters that best fit observed data. The generative model is the joint probability distribution $pr_\theta(x, z) = pr_\theta(x|z)pr_\theta(z)$ where $pr_\theta(z) = \mathcal{N}(z; 0, I)$ and the decoder is usually $pr_\theta(x|z) = \mathcal{N}(x; \mu, \sigma)$ with μ and σ being neural network estimators [41]. The inference network or the encoder allows computing a posterior approximation given a particular data point (e.g., an observation of a game). In particular, we use *amortized variational inference* that shares parameters overall observed data points [41]. The posterior approximation is defined as $p(z|x) = \mathcal{N}(z; \mu, \sigma)$ following the same principles as for the decoder. Parameter learning is performed using **E**vidence **LO**wer **B**ound (ELBO), which consists of a reconstruction term and a regularization term (Equation 24 and 26) where the first term encourages good reconstruction and the second term aims to model the data similarly to the prior (e.g., a Gaussian distribution). Intuitively, the aim is to create a generative model, meaning it is possible to reconstruct belief data over unseen data given the best estimates of data seen thus far. This fits well with a model in RL but requires adjustments to account for temporal dimensions. Motivated by the challenges of posterior collapse in VAE, Oord et al. proposed a categorical generative network, detailed in Section 5 [44].

• **State-Space Models** are particularly interesting because their background is founded in estimating trajectories such as the Apollo project in the '60s [42]. There are three types of SSMs, *filtering*, **prediction**, and *smoothing*, but we use prediction in this work and assumes the SSM model as a non-linear Gaussian function similar to VAE. The goal is to predict a future latent vector $p_\theta(z_t|b_{t-1}, a_{t-1})$ where b_{t-1} is the belief state and a_{t-1} is the action performed. Here we combine amortized variational inference from VAE's with learning a prediction model conditioned on belief states and actions from RL. [3]

• **Recurrent Neural Networks** are interesting because they aim to learn data dependencies stretched through time. While we have tested Liquid-Time Constant (LTC) networks and Gated Recurrent Units (GRU), we found that Long Short-Term Memory (LSTM) networks perform better in several tested environments. Following the theory of Belief MDP, we can learn a belief distribution with sufficient statistics of observations $b_t \equiv o_{1...t}$. We use an LSTM layer in our generative network to produce belief states b_t dependant on previous belief state b_{t-1} and action a_{t-1} but only use the notation z_t after stochasticity is added to the LSTM prediction [4].

• **To summarize**, the proposed predictive dynamics model is a probabilistic state-space model that using LSTMs to learn the belief distribution that parameterizes a Gaussian prior, learned using amortized variational inference from VAE's.

[3]We recommend [41] for further reading of VAE and [45] for SSM's.

[4]The belief-state is deterministic from the LSTM, and we found the latent-space variable much better if modeling as a Gaussian distribution

2.6. Q-Learning

Arguably the most central algorithm for fundamental RL is the Q-Learning algorithm [46]. Q-Learning is a value-based algorithm and uses the Q-function from Equation 9 as the basis for updating the policy parameter. Q-Learning is a model-free algorithm it meaning that it works independently of the underlying MDP dynamics $\mathcal{T}$ and is an off-policy algorithm, meaning that it can learn from samples collected by other policies. The Q-Learning algorithm follows the Bellman equations (9), where

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha\delta_t, \tag{12}$$

where $0 \leq \alpha \leq 1.0$ is the learning rate, and δ_t is the Bellman residual,

$$\delta_t = R_{t+1} + \gamma \underbrace{\max_{a \in \mathcal{A}} Q(s_{t+1}, a)}_{\text{off-policy}} - Q(s_t, a_t). \tag{13}$$

The Bellman residual δ_t denotes the temporal-difference (TD) error between current and subsequent state estimates. This procedure is named bootstrapping, e.g., we update our estimates of $Q(s_t, a_t)$ with another estimation $\max_{a \in \mathcal{A}} Q(s', a)$ where Q-Learning assumes best action is taken given current knowledge [47]. The max operator assumes that all future actions are optimal, which allows other policies to make decisions in an off-policy manner. It is also possible to derive another popular algorithm, SARSA (state-action-reward-state-action), by omitting the max operator such that the bellman residual,

$$\delta_t^{SARSA} = R_{t+1} + \gamma \underbrace{Q(s_{t+1}, a_{t+1})}_{\text{on-policy}} - Q(s_t, a_t) \tag{14}$$

is an on-policy TD update instead. The Q-Values are usually stored in a table in computer memory. However, it becomes infeasible to use traditional RL in larger problems because the algorithms store information in tables in the computer memory. For this reason, we use function approximators to learn a latent representation of the state-action value table.

2.7. Deep Q-Networks

Since traditional RL relies on tables to store parameters, it becomes impossible to assume an exact function to solve our policy optimization problem [2]. Instead, function approximation and specifically using neural networks is an appealing approach as they have demonstrated the capability to learn high-dimensional functions [16]. Deep Q-Networks tries to estimate the Q-table so that $Q(s, a; \theta) \approx Q*(s, a)$. Learning of the parameters θ is done through minimizing the following loss-objective,

$$\mathcal{L}(\theta_i) = \mathbb{E}_{s,a,r,s' \sim \mathcal{P}(.)} \left[(\delta_i | \theta_i)^2 \right], \tag{15}$$

where δ_i is the bellman residual from Equation 13. The first term is the reward, the second term is the greedy estimation from the target Q-network, and the third term is the inference Q-network.

2.8. Policy Gradient Algorithms

In contrast to value-based methods, **Policy Gradient (PG)** algorithms aim to find an optimal behavior policy through direct policy search. The policy is defined as a parametrized function with respect to θ and computes gradients based on an objective function,

$$\begin{aligned} J(\theta) &= \sum_{s \in \mathcal{S}} d_{\pi_\theta}(s) v_{\pi_\theta}(s) \\ &= \sum_{s \in \mathcal{S}} d_{\pi_\theta}(s) \sum_{a \in \mathcal{A}} \pi_\theta(a|s) Q_{\pi_\theta}(s,a) \\ &\propto \mathbb{E}_{s,a,r,s' \sim \mathcal{P}(.)} \left[\ln \pi_\theta(a|s) Q_{\pi_\theta}(s,a) \right] \end{aligned} \tag{16}$$

where $d_{\pi_\theta}(s)$ denotes the stationary distribution for π_θ. [48]. Updates are performed using *gradient ascent*, that is, we wish to find parameters θ for π_θ that yields the highest return, written as $\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$. The gradient theorem [2] find that the gradient respect to θ of the objective function $J(\theta)$ is,

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}[Q_{\pi_\theta}(s,a) \nabla_\theta \ln \pi_\theta(a|s)] \tag{17}$$

where $Q_{\pi_\theta}(s,a)$ is interchangeable with any return based function (e.g., advantage function) and $\mathbb{E}_{\pi_\theta}$ indicates the empirical average over a finite set of samples, sampled using the algorithm [49]. Many algorithms build on the policy gradient framework and perhaps most notably the Proximal Policy Optimization (PPO) for its substantial empirical performance and simplicity. PG algorithms are notoriously difficult to train because estimates have high variance and no bias. Vanilla PG is perhaps most known for this behavior, making the algorithm more susceptible to local optima and perform poorly across larger state spaces. Many algorithms build on the policy gradient framework, and perhaps the most promising direction is the Proximal Policy Optimization (PPO) family for its substantial empirical performance and simplicity [50]. We define the ratio $ro_t(\theta)$ between current policy $\pi_\theta(a_t|s_t)$ and previous policy $\pi_{\theta_{old}}(a_t|s_t)$ such that $ro_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}}$. Substituting $\ln \pi_\theta(a|s)$ in Equation 16, the objective becomes,

$$J^{CPI}(\theta) = \mathbb{E}_{s,a,r,s' \sim \mathcal{P}(.)} \left[ro(\theta) Q_{\pi_\theta}(s,a) \right], \tag{18}$$

which is the Trust Region Policy Optimization (TRPO) objective in [51]. The problem with J^{CPI} is that maximization leads to excessively large policy updates, hence learning becomes unstable. The work on PPO propose a clipping Scheme to reduce the size of policy updates,

$$J^{CLIP}(\theta) = \mathbb{E}_{s,a,r,s' \sim \mathcal{P}(.)} \left[\min(ro(\theta) Q_{\pi_\theta}(s,a), \text{clip}(ro(\theta), 1-\epsilon, 1+\epsilon) Q_{\pi_\theta}(s,a)) \right], \tag{19}$$

where ϵ is a hyperparameter $0 \leq \epsilon \leq 1$, typically set in the range of $\epsilon \approx 0.2$.

2.9. Risk-Aware RL

From a traditional RL view, most algorithms are considered risk-neutral, as clearly seen in the update Equation for Q-Learning (Equation 9) and its deep learning counterpart (Equation 15.) The problem with risk-neutrality is that for safety-critical systems such as the Deep Warehouse simulator, traditional algorithms fail to learn without relying on experience from catastrophic states [20]. This work draws parallels to a safety-critical system using the Deep Warehouse and Deep RTS Lava environment[5] mini-game (Section 6), where the goal is to retrieve gold without entering catastrophic lava states.

• **Uncertainty.** One approach to safe RL is to quantify the *epistemic* or *aleatoric* uncertainty and define it as a notion of *risk*. Aleatoric uncertainty stems from the observations done of the environment. An environment might have some inherent noise and stochasticity that is not controllable by the agent, which further amplifies belief MDP (however, the belief distribution is epistemic uncertainty) and POMDP's because of added uncertainty beyond the observable state-space. Generally, we think of aleatoric uncertainty as something that is not changeable but quantifiable. On the other hand, epistemic uncertainty is the model uncertainty, or in other words, the uncertainty of whether our prediction is correct with current knowledge. However, epistemic uncertainty is learnable and is reduced as the model becomes more certain of predictions. [52] In a stochastic process such as an MDP, several measurable uncertainties exist, such as model uncertainty, reward uncertainty, and value uncertainty [53]. **Model uncertainty** considers the uncertainty in transition function $\mathcal{P}$ of an MDP and has successfully guided agents during learning [54]. This article only consider MDP's where the transition function is unknown, and hence, we must estimate the model, which adds even more uncertainty to the model. **Reward uncertainty** defines the uncertainty of receiving a specific reward r_t at state s_t given the action made. Most notably is the intrinsic motivation from [55] that defines an auxiliary reward signal from the extrinsic reward function $\mathcal{R}$. Work from [56] similarly proposes a dynamics-based prediction error that is used as a secondary reward signal. **Value uncertainty** considers the value function $v_\pi(s)$ as a source of risk. Due to the recursive nature of the Bellman equations (Equation 8), the value-function estimates increasingly accumulate errors as time $t \to \infty$.

• **Risk-directed exploration.** Following the work of [57], risk-directed exploration is an auxiliary signal that guides action selection in RL algorithms. We quantify exploration risk $\Psi : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$,

$$\Psi(s,a) = w\mathrm{H} - (1-w)\frac{\mathbb{E}[\mathcal{R}(s,a)]}{\max\limits_{a \in A} |\mathbb{E}[\mathcal{R}(s,a)]|}, \tag{20}$$

where H is the entropy,

$$\mathrm{H}(s,a) = -\hat{\mathcal{T}}(s'|s,a) \log \hat{\mathcal{T}}(s'|s,a). \tag{21}$$

The entropy H of a stochastic process is a measurement uncertainty that suits well for quantifying risk. The risk is denoted Ψ and, as seen in Equation 20, is used as a trade-off between normalized expected return

[5]The Deep RTS Lava mini-game can be found at `https://git.io/JzpnJ`

and system entropy [58]. The risk is weighted $0 \leq w \leq 1$ where $w \geq .5$ values the stochastic nature of the system more as a notion of risk. Furthermore, we define a utility function $\mathcal{U} : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$,

$$\mathcal{U}(s, a) = \rho(1 - \Psi(s, a)) + (1 - \rho)\pi(a|s), \tag{22}$$

where $\leq \rho \leq 1$ controls the risk-awareness of the agent. For $\rho = 0$, risk awareness is disabled, and as $\rho \rightarrow 1$, the agent becomes increasingly aware of risk during decision making. Note that the utility function $\mathcal{U}$ is policy agnostic and works with policy-based and value-based methods, and is usable with sampling techniques such as ϵ-greedy and as a Boltzmann distribution (softmax). [57, 59]

• **Risk-Sensitive RL** refers to the branch of safe RL, which expresses a balance between a weighted risk and the return,

$$\max_{\pi \in \Pi}(\mathbb{E}_{\pi}(\mathcal{R}(s, a)) - \beta\omega). \tag{23}$$

The first term of Equation 23 is the expectation of the return (Equation 3), and the second term is the weight $0 \leq \beta \leq 1$ of the risk-function $\omega : \mathcal{R} \times \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ (note that this is omega ω not to be confused with w in Equation 20) [60]. Literature has studied different definitions of risk, such as using uncertainty from the TD-Error [61, 62], reward uncertainty [63], and using a set of error-states [60]. We use two sources of uncertainty in the system, particularly the dynamics model entropy and the variance of a set n predicted rewards, similar to [63]. This article follows the work of [64] but uses the measured variance as a risk signal for a more risk-averse agent. We combine risk-sensitive RL with Equation 20, where safety is adjusted long term with exploration [57] and short term through risk-averse weighting of the return function. We detail further our particular approach in Section 5.

• **Goal-directed RL** is not directly a technique for reducing risk but has appealing properties that reduce the probability of entering catastrophic states and hence, is used in literature towards risk reduction [65]. Goal-directed reinforcement learning (GDRL) separates the learning into two phases, where phase one aims to solve the goal-directed exploration problem (GDE). To solve the GDE problem, the agent must determine at least one viable path from the initial state to the goal state. In phase two, the agent uses the learned path to find a near-optimal path. The two phases iterate until the agent policy is converged. [66] The modeling task is to compute costs using neural network approximators that follow an *episodic* training Scheme. During exploration, the algorithm records a buffer of visited states, and at the time the agent enters a terminal state, the buffer is labeled with the corresponding Euclidean distance from the goal. The training is a supervised learning problem, and if enough data is collected, the estimator can accurately predict the distance between the current state and the terminal state. We denote the cost $c_t \in C$ in Section 5 as the *normalized* distance from the current observed state to the goal state.

3. Related Work

In the majority of established systems in the industry, an expert system made from human reasoning acts as the controller for the environment [67]. It is critical for safe and stable learning in real-world environments so that ongoing operations are not interrupted, and this Section details related work that aims towards improving and solving safe model-based RL (SMBRL).

• **Lyapunov-functions**. Perhaps the most notable recent work in solving SMBRL problems is the work of Berkenkamp et al. proposing a region of attraction-based method that is guaranteed safe constrained to a set of safe-states. The author presents a learning algorithm with two assumptions (1) the model is Lipschitz continuous, and (2) a reliable and well-calibrated statistical model exists to allow accurate exploration, close to the ground truth model function (the environment). The author adopts the region of attraction from control theory and uses Lyapunov functions that constrain the problem to an invariant set of the policy space. Furthermore, the author finds that a Lyapunov function is derivable from the value function $v_\pi(s)$, given that all rewards are positive. The experiments and theoretical justifications demonstrate that the algorithm functions well for the inverted pendulum environment and that their proposed algorithm improves performance while also holding safety constraints. [68]. Chow et al. similarly use Lyapunov functions to solve constrained MDP (CMDP) problems with the assumption of a feasible baseline policy. The author proposes updating the Lyapunov function using bootstrapping and showing that the method integrates well with Q-Learning. The author does not provide convergence proof to the optimal policy but empirically demonstrates that their approach performs better than Lagrangian methods. [69].

• **Barrier Functions.** Similar to Lyapunov-based approaches, the use of barrier function aims to constrain the policy-space to a *safe-set* of possible policies that are guaranteed to work safely. Cheng et al. propose a combination of model-based dynamics learning, shielded RL, and model-free algorithms as actors. Barrier functions are forward invariant, similar to Lyapunov functions, and uses a temperature hyperparameter to determine how constrained the policy space is. The method guarantees safety. However, it builds on the assumption that a determined set of safety policies are given before training. The authors demonstrate that their method outperforms novel model-free algorithms in sample efficiency and safety but at the cost of return performance. [70]. In a similar direction, Yang et al. propose a novel actor-critic barrier function structure for multi-agent safety-critical systems. Specifically, the authors propose a two-player game architecture for guaranteed safety during learning and solve a known model with safety guarantees [71].

• **Human Intervention.** One of the fundamental questions to raise in a safety-critical environment is *when to trust your model*. In earlier work on Human Intervention techniques, Saunders et al. propose a simple framework for training an algorithm safely using human intervention for catastrophic states. For every timestep in the environment, a human participant evaluated the state and proposed action of the algorithm, and for catastrophic actions, the human corrects the agent and gives negative feedback. While this is set in a model-free context, the human can be considered a *corrected* model of the policy space, and the authors found their method to perform well but failed to scale due to the time used by the human. [72]. Similarly, Turchetta et al. propose a curriculum-based approach, Curriculum Induction for Safe Reinforcement Learning (CISR), which assumes a teacher that intervenes in the event of catastrophic states. The teacher policy guides the student, intervenes, and puts the student in safe states if the CMDP criteria are not met. The author describes the CISR framework as a meta-learning framework where the teacher policy is an optimizable hyperparameter. However, the algorithm makes assumptions that there exists an intervention set defined before learning. The authors demonstrate that the student policy performs significantly better in the Frozen Lake environment when erroneous states are intervened compared to no intervention. [73]

• **Summary.** Throughout the last decade, many different approaches have been proposed toward improving safety in RL algorithms. While this article is out of scope to describe all methods, this Section describes promising directions in safe RL. This article draws inspiration from several other works in safe RL, which the respective authors best described in [57, 60, 61, 62, 64, 65, 53, 74], but otherwise referred to throughout this work. Furthermore, a brief overview of traditional safe-RL work is described in [4].

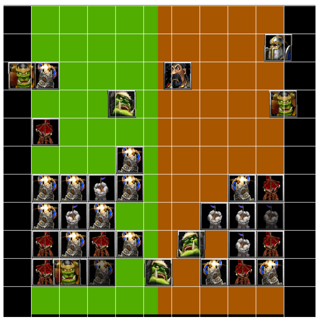
4. Environments

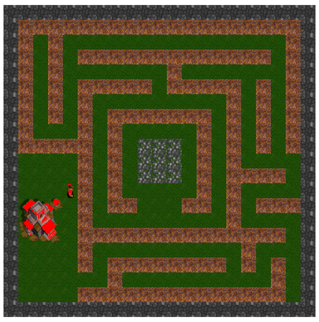
In the extreme pace that RL is moving, it is essential to use benchmarks that provide trivially replicable environment conditions. It is also important to have flexible benchmarks and can grow alongside the progress of research. Game-Complexity is a metric to determine the difficulty of a task, and while it does not account for uncertainty or stochasticity in the environment, it is often in literature. For example, Chess has state-complexity of $\sim 10^{50}$, Go (19x19) = $\sim 10^{360}$, and for StarCraft II, some estimations range from 10^{1685} to 10^{36000} [78]. As iterated, there is a clear gap in the literature of experimental environments, and the goal is that efforts such as MicroRTS [22], ELF [76], along with Deep Line Wars [75], Deep Warehouse [20], and Deep RTS [23] help fill the gap. In particular, all of these environments are flexible in that they allow to design of mini-games of nearly any state-complexity and allow configuration that tests the safety of algorithms. This Section thoroughly presents the experimental environments used in Section 6 and outlines the evaluation setup for each environment.

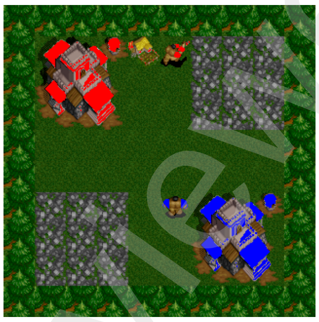
4.1. Deep Line Wars

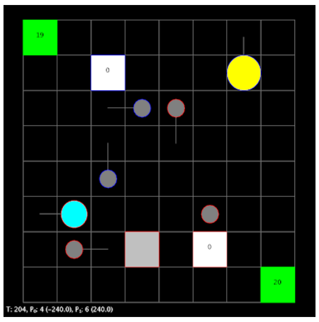
The Deep Line Wars (DLW) RTS environment provides a lightweight version of the Deep RTS game that offers a way to train algorithms on offensive and defensive strategies. Figure 4a illustrates the graphical observation, and to succeed in the DLW environment, the player should master a balance between (1) base construction, (2) economy management, (3) defensive planning, and (4) offensive evaluation.

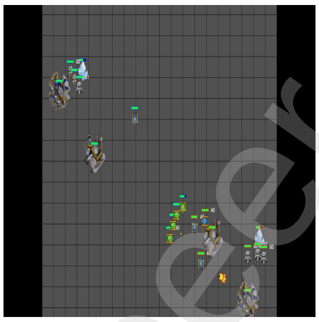
These objectives seem trivial to master individually but pose a significant challenge when combined. The DLW game is a two-player RTS game where the goal is to build a base and send units towards your

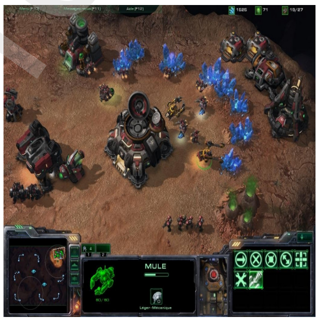

(a) Deep Line Wars [75]

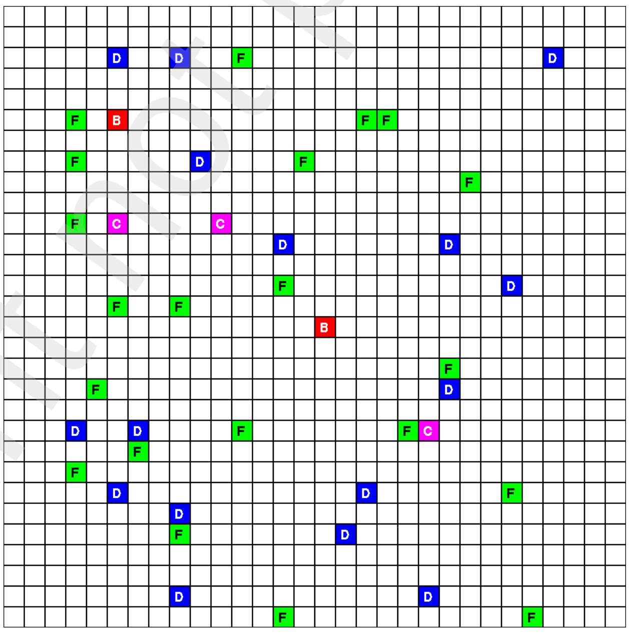

(b) Deep RTS Maze Objective [23]


(c) Deep RTS One Versus One [23]


(d) MicroRTS [22]


(e) ELF: Mini-RTS [76]


(f) StarCraft II Mini-Games [77]



(g) Deep Warehouse. In a simple cube-based ASRS system, the environment consists of (**B**) passive and (**C**) active delivery-points, (**D**) pickup-points, and (**F**) taxis.[20]

Table 2: MicroRTS maps with the corresponding map-size.

Map	Size
basesWorkers8x8A	8×8
basesWorkers16x16A	16×16
BWDistantResources32x32	32×32
(4)BloodBath.scmB	64×64
FourBasesWorkers8x8	8×8
TwoBasesBarracks16x16	16×16
NoWhereToRun9x8	9×8
DoubleGame24x24	24×24

opponent's base. DWL supports several modes from fully deterministic and observable state-space to partial observable stochastic state-spaces. The observations are available as a compact vector representation and represented as pixel images [75].

4.2. Deep RTS

The Deep RTS game environment enables research at different difficulty levels in planning, reasoning, and control. Deep RTS aims to narrow the research gap between Micro RTS [79] and StarCraft II [77]. Deep RTS compose games as scenarios that define a particular goal to win the game. For the most primitive scenario, the goal is to gather a set amount of gold before the agent receives a reward signal and the game terminates. The most complex environments support observability, environmental danger, and other opponents. It is possible to define delayed actions and delayed rewards to increase the difficulty of the task.

4.3. Micro RTS

Micro RTS is a simple RTS game designed to conduct AI research. The idea behind Micro RTS is to strip away the computational heavy game logic and graphics, seen in Figure 4d, to increase the performance and enable researchers to test theoretical concepts quickly [22]. The microRTS game logic is deterministic and includes options for fully and partially observable state spaces. The primary field of research in microRTS is game-tree search techniques such as variations of Monte-Carlo tree search and minimax [80, 22, 81] but is also used for deep learning techniques, especially in the larger maps, seen in Table 2.

4.4. ELF: Mini-RTS

The goal in Mini-RTS is for the agent to destroy the opponent's base with its troops. Players have units, resources, and a base and must balance economics for defensive and offensive planning. It is possible to

Table 3: List of mini-game maps in SC2LE

Map
MoveToBeacon
DefeatRoaches
BuildMarines
CollectMineralShards
CollectMineralAndGas
FindAndDefeatZerglings
DefeatBanelingsAndZerglings

expand the base with the worker unit type to build barracks and expand the offensive powers. The ELF game engine is tick-driven, meaning that the agent must make a decision per tick. Mini-RTS is partially observable due to the fog-of-war, and thus the agent is only presented with imperfect information. However, compared to StarCraft II, Mini-RTS is significantly less complex both logic-wise and graphically wise, seen in Figure 4e. The Mini-RTS environment features two built-in strategies; AI-Simple, AI-Hit-and-run where both are used in the experiments.

4.5. StarCraft II

SC2LE (StarCraft II Learning Environment) bridges the programming language Python and the StarCraft II state-space and action space. SC2LE is an initiative from Blizzard and DeepMind to demonstrate the capabilities of AI in RTS. StarCraft II is depicted in Figure 4f and is a complex environment that requires short and long-term planning. It is difficult to observe a correlation between actions and rewards due to the imperfect state information and delayed rewards, making StarCraft II one of the hardest challenges for RL algorithms to solve [77]. In addition to the full-game case, SC2LE features mini-games suitable for RTS research, as seen in Table 3.

4.6. Deep Warehouse

Training algorithms in real-world environments have severe safety challenges during training and suffer from low sampling speeds [82]. The Deep Warehouse environment features discrete and continuous action and state spaces. The environment has a wide range of configurations and is the only open-source implementation that aims to simulate proprietary automated storage and retrieval systems to the best of our knowledge[6].

In the context of warehousing, an Automated Storage and Retrieval System (ASRS) is a composition of computer programs working together to maximize the incoming and outcoming throughput of goods. Using

[6]The deep warehouse environment is open-source and freely available at `https://github.com/cair/deep-warehouse`

an ASRS system in logistics has many benefits, including high scalability, increased efficiency, reduced operating expenses, and operation safety. We consider a cube-based ASRS environment which can be thought of as a 3-dimensional asymmetrical rectangle with goods stored depth-wise. Taxi agents collect and deliver goods to delivery points on the rectangle's uppermost layer and usually work alongside other agents. A computer program controls the taxi agent that reads its sensory data to determine the following action. Although these systems are far better than manual labor warehousing, there is still significant improvement potential in the current state-of-the-art. Most ASRS systems are manually crafted expert systems, which due to the high complexity of the multi-agent ASRS systems, only performs sub-optimally. [83]. The Deep Warehouse experiments use a grid-size of 11×11 with 10 agents, 22×22 with 50 agents, and 41×41 with 100 agents, respectively.

4.7. Summary

Zooming out on the landscape of RL for RTS games, it is clear that the challenge is far from mastered when considering sustainability, safety, or flexibility. There is, without doubt, a progressive pace towards methods that perform better within specific environments, but as of yet, StarCraft remains the ultimate goal for sustainable RL algorithms [48, 79]. For the experiments in Section 6, we use the following environments:

- **Deep Line Wars**
  For the experiments, we use three map sizes in Section 6, 12×11 , 22×22, and 40×30, respectively, where the opponent uses the built-in *expert* strategy.

- **Deep RTS**
  In Section 6, we present results for two scenarios in Deep RTS where the first is a single-player objective to navigate through a maze and find gold, seen in Figure 4b. This environment is used to evaluate safety during training in RTS games. The second scenario, seen in Figure 4c is a one versus one, 10×10 map with four available units, four available buildings, and over 100 possible action combinations every game state update.

- **Deep Warehouse**
  To demonstrate safety improvement of S-ORACLE compared to fully model-free methods, we train the algorithms on three grids with various concurrent agents in each grid size. The experiments run in grid-sizes of 11×11, 22×22, and 41×41 with 10, 50, and 100 manhattan-distance based co-agents, respectively.

- **MicroRTS**
  We use the same environments as in the IEEE Conference of Games for recent and previous years, listed in Table 2

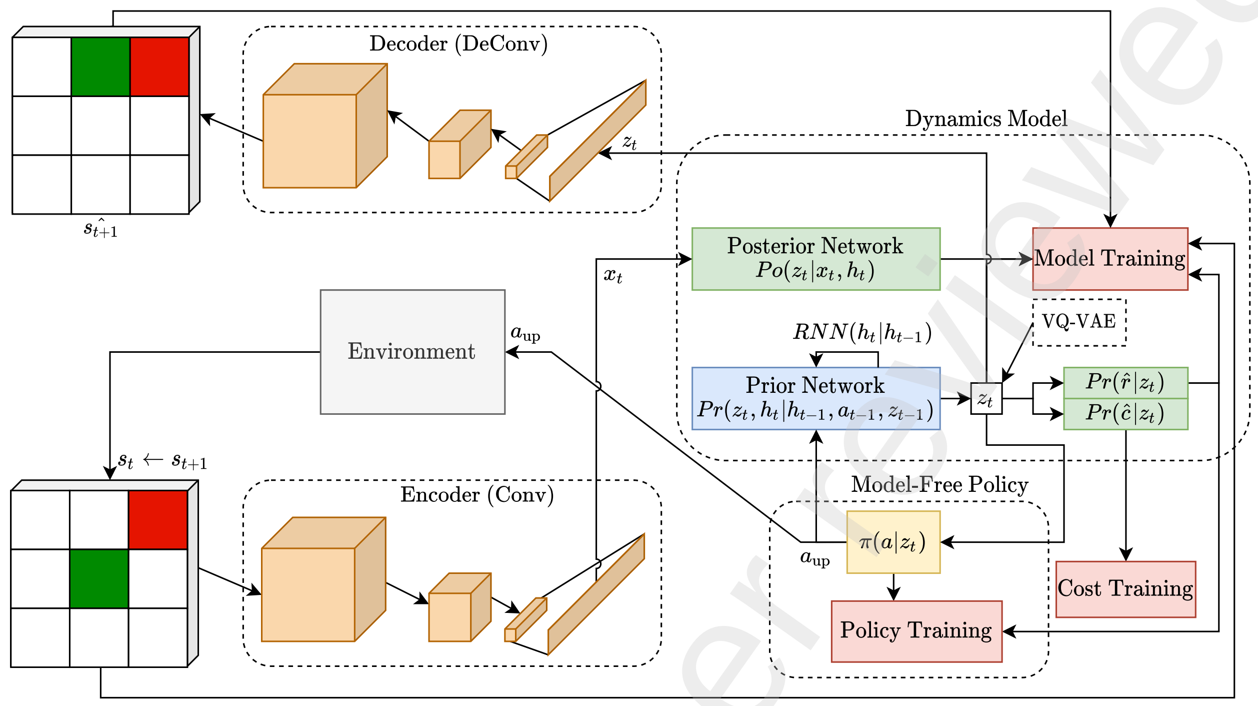


Figure 4: The S-ORACLE model. The environment produces an initial state s_t that feeds into a ConvNet. The flattened representation x_t is fed into the posterior distribution and produces a latent vector z_t. Concurrently, the prior distribution uses the previous hidden-state to predict a belief of the latent vector. The latent vector is the backbone for predicting a reward and the state-cost. The policy use trained parameters to predict an action and is sent to the environment. Every component of the model trains jointly in where each *training-block* (e.g., the colored squares) plays a part in the final optimization objective.

- **ELF: Mini-RTS**
  The Mini-RTS environment features two built-in strategies; AI-Simple, AI-Hit-and-run where both are used in the experiments.
- **Starcraft II**
  We use all mini-games available to SC2LE, which are listed in Table 3.

5. Learning dynamics for Planning and control

The S-ORACLE algorithm is a novel end-to-end architecture for training model-free algorithms on a dynamics model learned through observations of the true environment. S-ORACLE is a combination of state-of-the-art deep learning techniques; stochastic recurrent state-space models (SRSSM) [84], variational autoencoders (VAE) [41], and vector-quantization [85]. The model contains a deterministic encoder and a stochastic decoder, a stochastic dynamics model, and a policy. The full architecture is visualized in Figure 4 and the scope of this Section is to describe each component of the S-ORACLE model explain the desired outcome running model-free algorithms for safer planning and decision-making.

5.1. Deterministic Encoder and Stochastic Decoder

When the agent observes a high-dimensional input such as images, we must reduce dimensional to reduce computational complexity. The convolutional layers are interchangeable with fully connected layers for vector-based inputs, but we consider pixels for this article.

• **The decoder** is a deconvolution network that upsamples the compressed latent variables to the spatial structure of the input variable o_t. The last layer of convolutions predicts μ and σ that parameterize gaussian distributions for all output *pixels* in $\hat{o}_{t+1}$. Experiments revealed significant performance benefits of a stochastic decoder output because it added variability during training, making the predictions far more accurate than fully deterministic layers. There are no particular differences during inference, as we sample the mean, which roughly equates to the mean value of deterministic neural networks [84].

• **The encoder** is a three-layer deterministic convolutional neural network (CNN) that aims to capture information from pixel observations o_t and consequently reduce dimensionality before reshaping the compressed representation to a vector x_t[86]. The idea is that we compress observation to a compact representation, compute an internal state in the dynamics model, and decompress the internal representation. During training, the overall goal is for the dynamics model to resemble the same behavior as $\mathcal{T}(s'|s,a)$, where the model is denoted $\hat{\mathcal{T}}(s'|s,a)$. The encoder and decoder are learned using the MSE (cross-entropy) between the predicted observation $\hat{s}_{t+1}$, and the *after-the-fact* observation s_{t+1},

$$\mathcal{L}_{OBS} = -\mathbb{E}[logp(o|z)]. \tag{24}$$

5.2. Dynamics Model

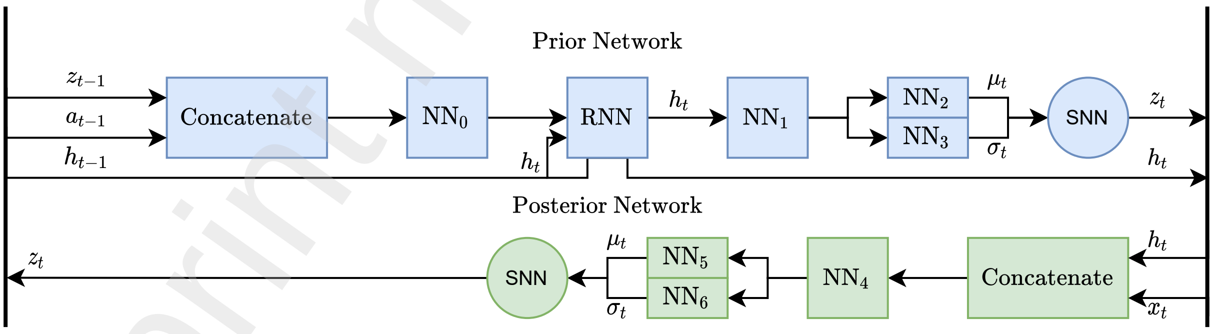


Figure 5: Detailed overview of the dynamics model forward-pass from Figure 4. The prior network uses last-step information, along with an action (the action that leads up to state s_t is denoted a_{t-1}), and outputs the computed hidden-state and the sampled latent-state variable z_t. For LSTM, we store the cell memory along with the hidden-state. The posterior network takes in the observed information x_t and our prior belief state h_t and predicts the *informed* latent-state variable z_t. Although both latent-state variables are denoted z_t, the prior latent-state variable is denoted $\hat{z}_t$ during training.

S-ORACLE is a model-based approach and attempts to estimate the true MDP transition function, described in Section 2.5. We use neural networks to estimate and combine VAE and SRSSM to create a highly expressive probabilistic model. S-ORACLE models two distributions, a prior model and a posterior model. Figure 5 depicts the forward pass of S-ORACLE after the observation o_t encodes to x_t.

• **The prior network** (generative network predicts the next state latent-space variables z_t using information from the last time-step; the latent-state z_{t-1}, current action a_{t-1}, and hidden-state h_{t-1}. Note that for LSTM, which is used in the experiments, the cell-state is preserved between inferences when using LSTM. The RNN node calculates the next hidden-state splits into μ and σ to parameterize Gaussians that predict z_t.

• **The posterior network** (inference model) concatenates hidden-state h_t and the encoded observation x_t and parametrizes Gaussian distributions similar to the prior network, which predicts *informed after-the-fact* latent-space variables.

• **The forward-pass** of the dynamics model is as follows where the prior model performs the following operations:

1. Compute $u_t = \text{concat}(z_{t-1}, a_{t-1})$
2. Compute RNN state $h_t = \text{RNN}(u_t)$
3. Parameterize mean $\mu_t = \text{NN}_2(h_t)$ diagonal covariance matrix $\sigma = \text{NN}_3(h_t)$
4. Sample from Gaussian distributions $\text{SNN}_\theta(z_t|h_t) \sim \mathcal{N}(h_t; \mu, \sigma)$

where all steps are performed for every sample and form our prior beliefs of the latent variables. The posterior model, seen in Equation 25b, depends on previous hidden-state h_{t-1}, action a_{t-1} including the encoded state-observation x_t. The posterior model can be summarized to the following procedure:

1. Compute $u_t = \text{concat}(h_t, x_t)$
2. Parameterize mean $\mu_t = \text{NN}_5(u_t)$ diagonal covariance matrix $\sigma = \text{NN}_6(u_t)$
3. Sample from Gaussian distributions $\text{SNN}_\psi(z_t|h_t) \sim \mathcal{N}(x_t; \mu, \sigma)$

• **Training** takes inspiration from previous work in [87] using variational inference [41] combined with Stochastic Recurrent State Space Models (SRSSM) from [84]. VAE and SRSSM are highly expressive model classes for learning patterns in time series data and system identification (e.g., learning dynamics model from observed data) [88]. We train the algorithm similarly to VAE's using amortized variational inference since $po_\theta(z|x) = \int_z \frac{po_\theta(x|z)po_\theta(z)}{po_\theta(x)} dz$ is intractable [89]. The generative model (prior) pr_θ and the inference model (posterior) po_θ is denoted,

$$\textbf{Prior Model}: pr_\theta(z_t, h_t|h_{t-1}, a_{t-1}) \tag{25a}$$

$$\textbf{Posterior Model}: po_\theta(z_t|x_t, h_t), \tag{25b}$$

where Equation 25a is the prior distribution that attempts to learn parameters θ that best fit the posterior distribution (Equation 25b) by minimizing the Kullback-Leibler (KL) distance. The intuition is that the posterior model learns dynamics of the MDP (environment) through observations x_t while the prior distribution must learn indirectly through optimization (parameters θ). To make the optimization tractable, we use the ELBO [41, 90] such that the optimization term for the SSM (generative and inference network) becomes,

$$\mathcal{L}_{SSM} = \mathcal{L}_{OBS} - D_{KL}[po_\theta(z|x)|pr_\theta(z)]. \tag{26}$$

5.3. Categorical Latents

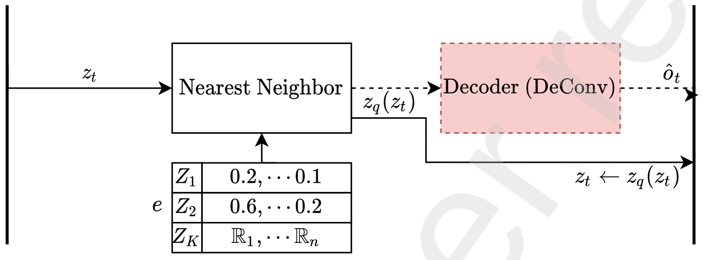


Figure 6: A detailed overview of the vector quantization network for mapping latent-space variables to categories after the generative network has predicted a continuous latent-space variable. The VQ-VAE module is depicted in Figure 4 and attempts to map variables from the generative network (prior) to categories. The dashed lines illustrate the process during training, while solid lines illustrate inference time computation. The algorithm finds an appropriate category using the nearest neighbor.

Following the work in [85], we use a variation of the VQ-VAE architecture, seen in Figure 6. However, we use it to categorize latent variables similar to [66] that allows for selecting policies to specific areas of the state-space and is a promising approach towards automating the options selected in the options framework [91]. Furthermore, VQ-VAE shows efficiency for planning and predictive learning [92] and does not suffer from posterior collapse. We observe that the dynamic model recovers from posterior collapse combined with stochastic weight averaging (SWA) combined with categorical latent motivate using VQ-VAE.

Inference and training is straightforward where the sampled latent-variables from the SSM mapped to a categorical codebook $Z_1 \cdots Z_K$ with K (hyperparameter, see Table B.18) possible categories of latent-space variables. The output is the category closest to the input, and we updated the latent variable $z_t \leftarrow z_t^k$. Training occurs jointly with rest of the model,

$$\mathcal{L}_{VQ} = ||sg\,[z_e] - e||_2^2 + \beta_{\text{vq}}||z_e - sg\,[e]\,||_2^2, \tag{27}$$

where e is the codebook. The first term is the codebook alignment loss which updates the selected category closer to the SSM latent vector. The sg denotes the stop-gradient operator, which prevents gradients computation for the SSM as this term only concerns updating the codebook. The second term moves the

SSM latent-vector towards the codebook but we wish to limit the influence on SSM, hence, $\beta_{\mathrm{vq}} \approx 0.05$ in contrast to [44] using $0.25 \leq \beta_{\mathrm{vq}} \leq 2.0$.

5.4. Rewards, Risks, and Costs

Using the learned dynamics model, we estimate rewards and costs as part of the feedback that fuels the learning of the RL agent. The feedback signal comprises regular rewards, uncertainty-based risk rewards, and costs that measure the distance between the current state and a positive terminal state. We follow the background in Section 2.9 and use Equation 23 to define the return function. The reward function $\hat{\mathcal{R}}$ is a parameterized Gaussian distribution learned using the squared loss between predicted and actual reward during training. The cost is learned separately after a terminal state is experienced by labeling previous states with the temporal difference (e.g., how many steps it took from state s_{t-n} until the terminal state s_t). The risk feedback is learned as a byproduct of the learned predictive reward model as we are interested in using uncertainty as a risk measurement $Risk = \mathrm{Var}(\hat{\mathcal{R}})$, which gives us the reward function,

$$\mathcal{R}_{oracle} = \mathcal{R}(z) - Var(\hat{\mathcal{R}}) + (1 - C(z)). \tag{28}$$

The S-ORACLE risk-aware approach summarizes in two steps for reducing risk during training and action inference. The first term in Equation 22 is the inference-time risk reduction, while during learning, the agent utilizes 22 for action selection and Equation 23 with $\omega = \mathrm{Var}(\hat{\mathcal{R}}(z))$. The second term is the Cost function $C(z))$ that predicts the normalized distance to a positive goal state. To optimize and learn the reward function and cost function, we minimize the following,

$$\mathcal{L}_{Rew} = \underbrace{\mathbb{E}[logpr(R|z)]}_{reward-loss} + \underbrace{sg(\mathbb{E}[logpr(C|z)])}_{cost-loss}. \tag{29}$$

5.5. Actor Policy Ensemble

The actor ensemble's primary objective is to make informed decisions that lead to safe trajectories to a goal state. In this work, S-ORACLE uses an ensemble of model-free approaches, specifically A2C, DQN, RAINBOW, IMPALA, and PPO. While all of the algorithms are different, they fundamentally share the objective of maximizing rewards which fuel the motivation for modeling the reward function in Equation 28. Specifically, all actor algorithms optimize towards the risk-aware reward function and empirically improve safety significantly. The actor ensemble is defined,

$$\pi_{ensemble} = \text{majority\_vote}(\{\pi_{A2C}(a|z), \pi_{DQN}(a|z), \pi_{RAINBOW}(a|z), \pi_{IMPALA}(a|z), \pi_{PPO}(a|z))\}, \tag{30}$$

where $\pi_{ensemble}$ is the majority voting policy that selects action a_t. Note that all policies select actions according to the risk-adjusted utility function in Equation 22 first proposed by [57], but using $\hat{\mathcal{R}}$ instead of the environment reward signal $\hat{R}$. Although the ensemble actor does not guarantee safety during training or learning, it shows a promising leap towards safer algorithms empirically, and Section 8 discusses the implications and possible methods to improve safety further.

5.6. *Tuning and Training S-ORACLE*

The S-ORACLE algorithm has many hyperparameters for tuning stability and performance, and is located in Table B.18. Experiments illustrate S-ORACLE robust between most environments but required additional hyperparameter tuning for StarCraft II and Deep RTS. LTC and GRU were tested but found LSTM to perform better for tested environments. Another notable hyperparameter choice enables adaptive gradient clipping (AGC), a novel approach to clip the gradient from historical norms [93]. Additionally, gradients are clipped between -100.0 and 100.0 to increase training stability. The S-ORACLE model optimizes all objectives jointly,

$$\mathcal{L}_{ORACLE} = \mathcal{L}_{OBS} + \mathcal{L}_{Rew} + \gamma\mathcal{L}_{SSM} + \mathcal{L}_{VQ}, \tag{31}$$

using **S**tochastic **W**eight **A**veraging (SWA) with the AdamW optimizer [94]. SWA is a novel approach to ensemble learning where the objective is to widen the optima space such that it is easier to find and to give a better generalization of the model [95][7].Compared to other ensemble learning techniques, SWA only requires a single model where snapshots are stored every n epochs that are averaged every m epochs. SWA has different learning rate strategies (e.g., cyclical learning rate), and a linear cyclical learning rate is chosen for S-ORACLE.

5.7. *Summary*

This Section summarizes S-ORACLE as a novel approach towards safer reinforcement learning in RTS and industry-near environments. In contrast to prior work, **no assumptions on prior knowledge** at the cost of *not guaranteeing safety* if left unsupervised. However, using an **expert system** significantly decreases the likelihood of entering catastrophic states if used to pre-train the dynamics model. Section 6 shows that the **actor ensemble performs well within a presumable safe state-space set, without external guidance** if pretraining of the dynamics model is allowed. The S-ORACLE is summarized to the following procedure:

1. Read state observation o_t
2. Extract features of the observation x_t
3. Compute latent-state using x_t for the prior and posterior $\hat{z}_t$
4. Compute categorical latent-state for latent-space using VQ-VAE architecture z_t
5. Predict using latent-state z_t
    - Decode latent-state in decoder and sample possible future observation $\hat{o}_t$
    - Predict action for policy ensemble given latent-state using majority voting $\pi_{ensemble}(a_t|z_t)$

[7]The author further demonstrated effectiveness in *Improving Stability in Deep Reinforcement Learning with Weight Averaging* but the work is not peer-reviewed. However similar results are demonstrated in [96]

- Predict reward of the latent-state $\hat{r}_t$
- Predict cost of the separate model using latent-state $\hat{c}_t$

6. Compute gradients jointly following Equations 24, 26, 27,29, and separately for the policy ensemble and cost network. Use the Adam optimizer with SWA and gradient clipping.

6. Empirical Evaluation

This Section presents empirical results from the five RTS environments Deep RTS, ELF, MicroRTS, StarCraft II, and Deep Warehouse, for safety evaluations. The background for the environments is found in Section 4. Each environment is tested with five different model-free algorithms, A2C, DQN, RAINBOW, IMPALA, PPO using standard hyperparameters from the respective literature. Additionally, we test with the DVAE algorithm, a model-based approach that uses variational autoencoders to learn a dynamics model that is later used to train a PPO policy [96].

- **Experiment Setup.** The experiments are allocated one GPU per algorithm from a pool of 2x 2080 TI and 2x 1080 TI cards. Experiments allocated with a 1080 TI are given 37% longer train time to compensate for slower training speeds. For climate footprint estimations, we limit the GPU power draw to 250w and divide the total energy consumption of the 1080 TI-based experiments to compensate for the additional time. Model-free algorithms train for 24 hours and model-based trains for 16 hours to demonstrate efficiency. Performance is tested for several episodes, where the number of episodes is specified per environment due to computational limitations.
- **Experiment Scope.** The goal of the **experiment is to demonstrate the raw performance and safety performance of S-ORACLE**. Concurrently, a baseline is constructed for future work. Experiments demonstrate that S-ORACLE generalizes well for multiple problems, although the algorithm still has improvement potential in behavior performance, sample efficiency, and safety. The results also systematically report S-ORACLE's energy and carbon footprint compared to other tested algorithms as suggested by Henderson et al. [6].

6.1. Safety Evaluation

S-ORACLE aims to learn an agent agnostic feedback signal that directs the risk-neutral agents towards safer trajectories during training. S-ORACLE learns in a two-objective process where the first objective is to learn the dynamics model and the second is to learn an actor ensemble (see Section 5). S-ORACLE offer two training schemes to training the predictive model:

1. **Scheme #1**. Train dynamics model and actor ensemble concurrently, balancing the exploration-exploitation dilemma for risk management.

2. **Scheme #2**. Train dynamics model using external knowledge, using existing expert systems known to operate safely in the environment and subsequently, train actor ensemble off-line using dynamics model before carefully evaluate safety in live systems. This approach is seen as successful in prior work [96, 20].

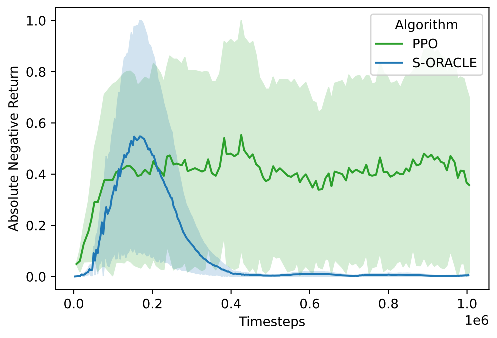
The first method is the most versatile because it does not require any knowledge *a priori* to solving the problem. However, the first method is at increased risk of entering catastrophic states. The second method improves empirical safety but relies on external systems to succeed. This Section aims to empirically demonstrate the safety performance of S-ORACLE compared to PPO in Scheme #1 and DVAE in Scheme #2. The experiments measure the ratio of incoming absolute negative returns to determine safety, normalized between $0.0 \leq x \leq 1.0$. For every 100 000 timesteps, the algorithm is evaluated for 100 episodes, and the measured min-max variance is illustrated using shades in the plots. S-ORACLE is evaluated in two environments using the described training schemes:

1. **Deep RTS** lava environment, seen in Figure 4b. The agent performs safe behavior when avoiding lava states. The goal is to reach the gold in the middle of the map.
2. **Deep Warehouse** logistics challenge. Transport goods from source to destination as fast as possible while avoiding collision with other agents. The algorithm controls a single agent, while the remaining agents use a manhattan distance-based heuristic. The agent is part of a larger logistics system with other taxi agents, depending on map size (see Section 4).

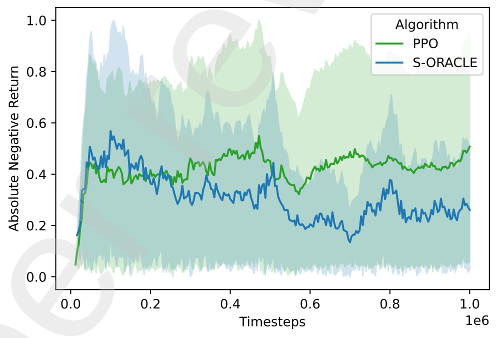
Using Scheme #1, the algorithms balance the exploration-exploitation dilemma during dynamics model training. Hence they are more susceptible to error states because of missing knowledge about the dynamics. Figure 7a-7d shows the safety violations ratio in Deep RTS and Deep Warehouse, respectively. In the Deep RTS Lava environment, it is clear that S-ORACLE learns to avoid lava, and for Deep Warehouse 11×11 (Figure 7b), the S-ORACLE gradually reduce the rate of negative returns compared to PPO. Similarly, for Deep Warehouse 22×22 and 41×41 (Figure 7c and Figure 7d), S-ORACLE accumulate fewer negative rewards, but the effect seems to diminish for more complex state-spaces. Collectively, the figures demonstrate that a risk-neutral algorithm (PPO) explores negative states at a relatively uniform rate, while the risk-averse agent (S-ORACLE) impacts the number of visited error-states significantly less.

In Scheme #2, the dynamics model is trained from observations of an expert system before training using the dynamics model occurs for the agent algorithm. The second approach demonstrated better safety awareness, naturally, because the algorithm learned much of the environment through the dynamics model before making actions live. Figure 8 shows the safety-awareness of S-ORACLE compared to DVAE, clearly showing that there is a downwards trend in the mean absolute negative return. However, as the environment becomes more complex (Figure 8d), the effect seems to diminish in contrast to Figure 8b.

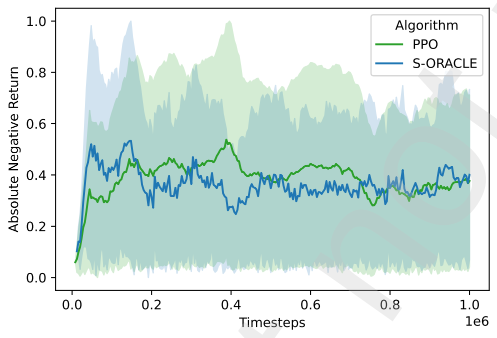
The empirical evidence clearly shows that S-ORACLE improves safety in Scheme #1 and Scheme #2



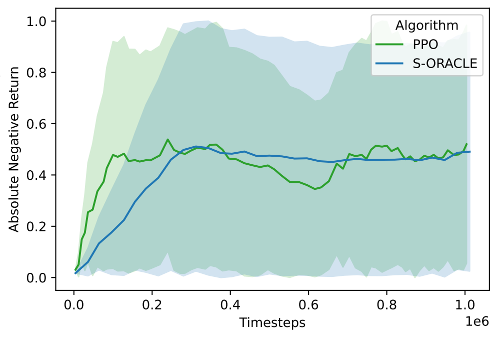
(a) Deep RTS Lava.



(b) Deep Warehouse 11 × 11.

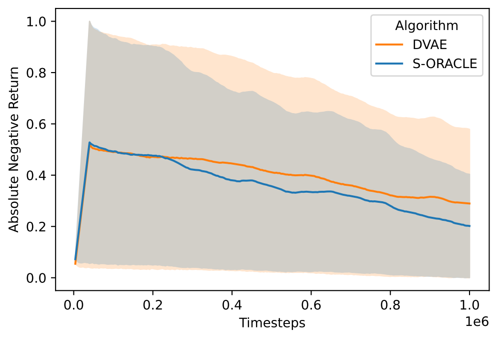


(c) Deep Warehouse 22 × 22.

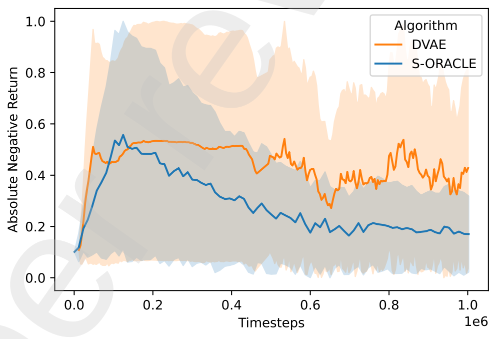


(d) Deep Warehouse 41 × 41.

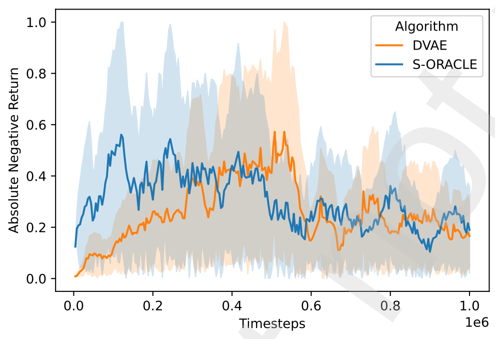
Figure 7: Safety violations rate following training Scheme #1. The y-axis is the mean absolute negative return where the absolute negative return is averaged per 10 000 timesteps.



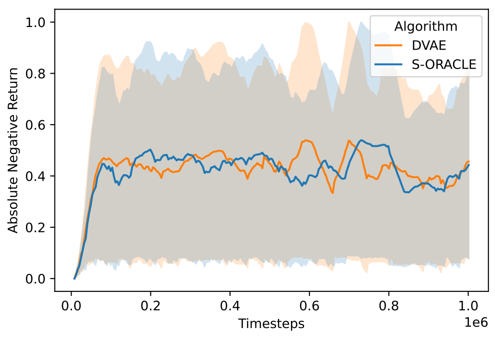
(a) Deep RTS Lava.



(b) Deep Warehouse 11×11.



(c) Deep Warehouse 22×22.



(d) Deep Warehouse 41×41.

Figure 8: Safety violations rate following training Scheme #2. The y-axis is the mean absolute negative return where the absolute negative return is averaged per 10 000 timesteps.

compared to model-free RL algorithms and the prior work of DVAE. Specifically, as the algorithm learns to navigate the environment, the agent receives fewer negative rewards than other tested environments.

6.2. Performance Evaluations

- **Deep Line Wars.** Table 4 shows the performance of the tested algorithms against the hard-coded expert agent. Each algorithm runs 1000 times and is averaged over. Specifically, we observe that all tested algorithms perform well, even for large maps. The S-ORACLE agent performs comparably to the tested model-free algorithms and outperforms all tested algorithms on average. The action space comprises 4 actions for cursor position, 4 build actions, and 4 spawn unit actions.
- **Deep RTS.** We evaluate the algorithms for 100 episodes in a maze-line problem as seen in Figure 4b and in a one-versus-one as in Figure 4c between the tested algorithms. For the maze-like environment, results and averaged. Table 5 shows the performance of tested algorithms in the maze-environment. All algorithms perform well, where the best algorithm, IMPALA, consistently found the best path after training. The other algorithms performed in the $\sim 95\% \pm 5$ range. Table 6 shows that S-ORACLE outperforms 5 out of 7 algorithms and scores comparably with PPO in 1v1 matches. We observed that S-ORACLE plays conservatively and wins by strategically blocking the opponent's path with houses during experiments. In the long run, the conservative behavior wins because the cost of units is more expensive than houses, and additionally, houses allow the player to build a larger army that trivially defeats the opponent with limited housing. The agent had access to 13 discrete actions at every timestep and received positive rewards for victories and zero during the game.
- **ELF: Mini-RTS.** The partial observability of Mini-RTS makes the learning task significantly harder to master compared to Deep RTS. Table 7 shows that S-ORACLE average 83% win ratio but had difficulties defeating the AI-Simple algorithm consistently. The Mini-RTS action space comprises 9 strategic actions in which the agent can construct complex strategies. We observe that the average-length game is approximately 3500 ticks where a positive reward is given for victories, a negative reward for defeats, and zero rewards in other states. Algorithms are tested for 100 episodes.
- **MicroRTS.** Perhaps the most widely used environment for competitions is the MicroRTS environment [22]. We test each algorithm for 100 episodes against strategies presented in the IEEE Conference on Games (COG-2019) competition. Table 8 shows the results, where each row illustrates the win rate against the column algorithm. We see that S-ORACLE outperforms the other algorithms, closely followed by DVAE, PPO, and IMPALA. The action space and state-space vary between maps but are thoroughly described in [97].
- **StarCraft II.** Perhaps the most compute-heavy experiment is the StarCraft II environments, albeit we only focus on mini-games of the original game. In particular, we use the mini-games from [77]

already standardized in literature, and we adopt these findings for our comparison. As seen in Table 9 no algorithm remains dominant in all environments, but the professional player from DeepMind (DM) [77]. However, the S-ORACLE algorithm scores on average better than the model-free approaches, likely because of the ensemble technique when voting for actions. The action space is challenging to implement because it comprises several hundred actions for every timestep. For this reason, we choose to adopt the same method as in [98].

Table 4: Deep Line Wars result. Each agent (row) plays against the built-in *expert* agent at different map-sizes (column). The last column is the average performance for all map-sizes. The cell values represent win ratio ranging from 0 to 1.

Algorithm | Map	12x11	22x22	40x30	64x64	Avg
A2C	1.00 ±0.0	0.96 ±0.01	0.86 ±0.07	0.69 ±0.3	0.88
DQN	1.00 ±0.0	0.92 ±0.03	0.89 ±0.1	0.85 ±0.08	0.92
RAINBOW	1.00 ±0.0	0.95 ±0.03	0.90 ±0.08	0.64 ±0.01	0.87
IMPALA	1.00 ±0.0	0.96 ±0.04	0.95 ±0.04	0.85 ±0.11	0.94
PPO	1.00 ±0.0	0.99 ±0.01	0.95 ±0.01	0.66 ±0.18	0.90
DVAE	1.00 ±0.0	0.99 ±0.01	0.89 ±0.1	0.77 ±0.05	0.91
S-ORACLE	1.00 ±0.0	0.98 ±0.01	0.97 ±0.01	0.87 ±0.07	**0.96**

Table 5: The results of the Deep RTS Maze environment. The environment as seen in 4b defines the reward function as $r_t = optimalRoute \times 2 - t$. The results are averaged over 100 episodes. The cell values represent accumulated score ranging from 0 to 113.

Algorithm	Score
A2C	91 ±5
DQN	98 ±9
RAINBOW	99 ±4
IMPALA	100 ±2
PPO	96 ±6
DVAE	97 ±11
S-ORACLE	94 ±5

6.3. Sustainability Report

Table 10 reports the contribution of $\frac{CO2}{kg}$ of the experiments following the work of Henderson et al. [6]. Algorithm performance is tested in 41 experiments and additional six safety experiments for PPO, DVAE,

Table 6: DeepRTS 1v1 results. The row is the win rate of the respective algorithm agains the column wise algorithm for 100 episodes. The cell values represent win ratio ranging from 0 to 1.

Algorithm	A2C	DQN	RAINBOW	IMPALA	PPO	DVAE	S-ORACLE	Avg
A2C	-	0.35 ±0.11	0.23 ±0.05	0.44 ±0.11	0.14 ±0.05	0.39 ±0.11	0.36 ±0.05	0.32
DQN	0.65 ±0.03	-	0.45 ±0.02	0.36 ±0.05	0.25 ±0.03	0.36 ±0.32	0.25 ±0.14	0.39
RAINBOW	0.77 ±0.02	0.55 ±0.15	-	0.59 ±0.05	0.47 ±0.07	0.55 ±0.05	0.45 ±0.02	0.56
IMPALA	0.56 ±0.07	0.64 ±0.11	0.41 ±0.08	-	0.45 ±0.04	0.59 ±0.22	0.36 ±0.18	0.50
PPO	0.86 ±0.07	0.75 ±0.01	0.53 ±0.11	0.55 ±0.02	-	0.52 ±0.05	0.56 ±0.03	**0.63**
DVAE	0.61 ±0.25	0.64 ±0.05	0.45 ±0.05	0.41 ±0.06	0.48 ±0.05	-	0.25 ±0.02	0.47
S-ORACLE	0.64 ±0.11	0.75 ±0.02	0.55 ±0.09	0.64 ±0.03	0.44 ±0.02	0.75 ±0.02	-	**0.63**

Table 7: ELF: Mini-RTS results. Each experiment runs for 100 episodes and is averaged over. The cell values represent win ratio ranging from 0 to 1.

Algorithm	AI-Simple	AI-Hit-and-run	Average
A2C	0.85 ±0.07	0.86 ±0.08	0.86
DQN	0.56 ±0.05	0.85 ±0.02	0.71
RAINBOW	0.75 ±0.11	0.88 ±0.05	0.82
IMPALA	0.66 ±0.02	0.96 ±0.09	0.81
PPO	0.78 ±0.04	0.97 ±0.11	**0.88**
DVAE	0.73 ±0.05	0.99 ±0.05	0.86
S-ORACLE	0.67 ±0.09	0.98 ±0.04	0.83

Table 8: MicroRTS averaged results for all tested mini-games. Results for individual environments are found in appendix Appendix A. The cell values represent win ratio ranging from 0 to 1.

Agorithm	RandomBiasedAI	NaiveMCTS	Tiamat	Droplet	UTS_Imass	Win Ratio
A2C	0.89 ±0.01	0.86 ±0.06	0.84 ±0.11	0.79 ±0.14	0.84 ±0.13	0.84
DQN	0.91 ±0.04	0.86 ±0.12	0.79 ±0.03	0.82 ±0.17	0.7 ±0.26	0.82
RAINBOW	0.89 ±0.1	0.88 ±0.06	0.83 ±0.08	0.79 ±0.21	0.79 ±0.11	0.83
IMPALA	0.88 ±0.05	0.86 ±0.09	0.86 ±0.05	0.87 ±0.06	0.85 ±0.12	0.86
PPO	0.91 ±0.02	0.92 ±0.02	0.83 ±0.07	0.84 ±0.05	0.83 ±0.07	0.86
DVAE	0.90 ±0.07	0.87 ±0.09	0.80 ±0.18	0.90 ±0.07	0.84 ±0.15	0.86
S-ORACLE	0.90 ±0.06	0.88 ±0.02	0.86 ±0.02	0.84 ±0.04	0.86 ±0.05	**0.87**

Table 9: StarCraft II results. The A2C, A3C, and DM are results from relevant literature, and the remainder is novel results in this work. We ran the experiments ten times and averaged the results. The cell values represent total accumulated return.

Environment	A2C [98]	A3C [99]	DM [77]	DQN	RAINBOW	IMPALA	PPO	ORACLE	S-ORACLE
MoveToBeacon	21.3	24	26	26	30	32	35	24	29
DefeatRoaches	72.5	47	41	100	81	91	75	60	77
BuildMarines	0.55	0.6	138	0	0	2	8	12	2
CollectMineralShards	81	45	133	3	12	41	53	55	58
CollectMineralAndGas	3320	371	6880	3978	3911	4251	4102	5212	5102
FindAndDefeatZerglings	22.1	25	46	45	21	23	19	29	35
DefeatBanelingsAndZerglings	56.8	43	729	62	20	423	251	305	530
Average Score	510.6	79.3	**1141.8**	602	582.1	694.7	649	813.8	**833.2**

Table 10: This work contributed 55.08 kg of CO2eq to the atmosphere, and used 1377.0 kWh of electricity. The columns describes the following from left to right: (1) Number of experiments performed, (2) Average hours of CPU time per experiment, (3) Average hours of GPU time per experiment, (4) Total CPU time for all experiments, (5) Total CPU time for all experiments, (6) Total kWh for all experiments, and finally, the total $\frac{CO2}{kg}$ contribution of the experiments.

Algorithm	Num Exp	Avg CPU H.	Avg GPU H.	Tot CPU H.	Tot GPU H.	Tot. kWh	$\frac{CO2}{kg}$
A2C	41	13.53	21.41	554.73	877.81	233.04	9.32
DQN	41	11.71	20.22	480.11	829.02	219.02	8.76
RAINBOW	41	19.24	15.74	788.84	645.34	180.66	7.23
IMPALA	41	15.55	14.55	637.55	596.55	164.76	6.59
PPO	45	16.22	19.22	729.90	864.90	234.11	9.36
DVAE	45	14.33	11.24	644.85	505.80	142.25	5.69
S-ORACLE	49	13.11	12.53	642.39	613.97	169.23	6.77

and S-ORACLE. Note that the model-based approaches have a training budget of 16 hours, and model-free algorithms have 24 hours. The Figure shows that model-based approaches utilize time significantly more, considering that total CPU and GPU time is close to model-free algorithms. The reason for better utilization is that model-based RL algorithms better saturate the GPUs when training using the dynamics model.

7. Discussion

Performance is not the dominant factor in safety-critical systems. However, the *niceness* of performing superhuman or beyond existing expert systems is appealing if the algorithm can provide safety guarantees. S-ORACLE cannot provide safety guarantees, but it provides empirical evidence of acceptable performance while lowering erroneous agent decisions. The algorithm demonstrates that, although following a generic and noise-inducing risk-averse reward signal, the algorithm demonstrates that good policies are possible to obtain and to perform adequately in most tested environments.

Safety is perhaps the most challenging trait to learn in an RL setting because it is not naturally present in the foundation of which the framework builds. Perhaps the first hint is found in the word *reinforcement* and the most understanding of the concept is to *build upon something* or *the process of encouraging or establishing a belief or pattern of behavior.* The RL literature often depicts the analogy of a child in a behavioristic way that through sensory stimuli such as vision, hearing, smell, feeling, and taste, the child learns through trial and error. However, to compare RL algorithms to a child, we must strip away all its sensory information and memory and feed it noisy and sparse information about the world. If so, the child (algorithm) has very few sensory sensations, and at the beginning of time t do not know the environment, it seems impossible to learn anything without trial and error. Or hence, the child analogy is inadequate. Drawing parallels to S-ORACLE, it is only natural that the algorithm performs erroneous decisions during learning. While there are methods to provide true safety guarantees via prior information and policy constraints [68], no method exists for learning a task perfectly without balancing the exploration-exploitation dilemma. Drawing further parallels to the mission-critical industry setting, One approach to **alleviate** this problem is to use existing algorithms or expert systems operational in the environment to build a world model accurate enough to guide model-free algorithms towards safe policies. As seen in the experiments using expert-system knowledge to train the dynamics model works empirically well in the Deep Warehouse environment. S-ORACLE cannot theoretically guarantee safety but empirically shows that it improves safety significantly compared to risk-neutral approaches during learning and inference. Combined with generalizable self-tuning constraints, the hope is that future work brings such work to allow fully safe decision-making in safety-critical systems.

8. Conclusion and Future Directions

This work investigates whether the safer model-based reinforcement learning algorithm S-ORACLE has (1) better performance, (2) sample efficient, (3) more sustainable, and (4) safer than traditional model-free deep RL algorithms. Based on experiments in Section 6, empirical evidence suggests that 2-4 hold except for 1, where the algorithm is on-par or inferior to model-free approaches.[8] This Section details further 1 to 4 and concludes the findings of this work.

[8]Similar conclusions are found in [100] emphasizing having smaller rollout horizons for better model prediction.

Performance is evaluated through the empirical score the algorithm obtained during experiments. Generally, we find S-ORACLE less performant and to have a larger spread in obtained scores. While S-ORACLE is not performing better, it closely follows in most experiments, although there is still work that needs addressing lesser performance. Our analysis is that the agent performs more conservative action sequences, which leads to defeat as the opponent gain territory dominance in the RTS games. However, in Deep Warehouse, we observe that the algorithm is on par with PPO and DQN, which is a good measurement of how performance is in industry-like environments (e.g., stationary environments). One possible solution would also allow for longer training, but at the cost of a less climate-sustainable model.

Sample Efficiency is significantly better than model-free algorithms, as naturally expected of a model-based approach. However, we do not S-ORACLE as a high sample efficient algorithm. For instance, we expect work such as [101] to have better sample efficiency. Our findings in Section 6 indicate that the model prediction error closely follows the sample efficiency. For instance, the StarCraft II environment is notoriously difficult to learn, and hence, it takes far longer to obtain a good model. In future work, we would like to improve the prediction model by using discrete latent space techniques more closely to work in [101].

Sustainability is an increasingly important topic in machine learning because most state-of-the-art algorithms in the literature have a high computational cost which has an immediate impact on global warming. For this reason, we have thoroughly evaluated our model through the lens of sustainability, that is, how much power draw our model takes to train. We follow the work of [6], and while this approximation is far from perfect, it is a good indication of the overall sustainability of the model. Unfortunately, we could not find any model that reports CO2 emissions, but we hope that our work serves as a good baseline for future work in safe model-based RL.

Safety is crucial for an algorithm that aims to reach industry-like production environments outside the research lab, and by no means are S-ORACLE the answer to a perfectly safe algorithm, but rather towards a more safe alternative. In our experiments, we quantify the occurrences of catastrophic states during learning, and we observe that S-ORACLE has significantly fewer error-states visited than fully model-free approaches. Our vision is for an algorithm without *assumptions built on assumptions* that work in practice and can scale to more complex environments. To the best of our knowledge, there is no similar work in safe RL for complex games such as RTS games, and we hope that future work may include impressive progress towards more safety-aware agents than S-ORACLE.

Summary We present a novel, safer model-based RL agent for RTS games and industry-like applications. We empirically show that it performs well while maintaining better safety decision-making and is more sustainable than model-free approaches.

Future work will attempt to address several shortcomings in our model and simplify the model where possible, similar to the works on TRPO and PPO. One particular problem is that the algorithm is susceptible to posterior collapse during training, well known in variational inference. Specifically, [102] is an appealing approach to increase stability, which we aim to address in future work. Lastly, we aim to reduce the number of model hyperparameters resulting from immense testing of theoretically justified methods. However, some hyperparameters are found to work only for particular values. Hence it is sensible to better define hyperparameters in closed sets compared to definitions for any real number, as seen in B.18

9. Acknowledgements

We would like to take this opportunity to thank the reviewers for their immense contribution to evaluate this contribution. Also, we thank the editors for their interest in producing a special issue on risk-aware autonomous systems for artificial intelligence.

References

[1] R. Bellman, A Markovian Decision Process, Journal of Mathematics and Mechanics 6 (5) (1957) 6. doi:10.1512/iumj.1957.6.56038.
URL https://www.jstor.org/stable/24900506

[2] R. S. Sutton, A. G. Barto, Reinforcement learning: An introduction, 2nd Edition, A Bradford Book, Cambridge, MA, USA, 2018.
URL https://dl.acm.org/doi/book/10.5555/3312046

[3] R. S. Sutton, Dyna, an integrated architecture for learning, planning, and reacting, ACM SIGART Bulletin 2 (4) (1991) 160–163. doi:10.1145/122344.122377.
URL https://dl.acm.org/doi/10.1145/122344.122377

[4] J. García, F. Fernández, A Comprehensive Survey on Safe Reinforcement Learning, Journal of Machine Learning Research 16 (2015) 1437–1480.

[5] O. Vinyals, I. Babuschkin, W. M. Czarnecki, M. Mathieu, A. Dudzik, J. Chung, D. H. Choi, R. Powell, T. Ewalds, P. Georgiev, J. Oh, D. Horgan, M. Kroiss, I. Danihelka, A. Huang, L. Sifre, T. Cai, J. P. Agapiou, M. Jaderberg, A. S. Vezhnevets, R. Leblond, T. Pohlen, V. Dalibard, D. Budden, Y. Sulsky, J. Molloy, T. L. Paine, C. Gulcehre, Z. Wang, T. Pfaff, Y. Wu, R. Ring, D. Yogatama, D. Wünsch, K. McKinney, O. Smith, T. Schaul, T. Lillicrap, K. Kavukcuoglu, D. Hassabis, C. Apps, D. Silver, Grandmaster level in StarCraft II using multi-agent reinforcement learning, Nature 2019 575:7782 575 (7782) (2019) 350–354. doi:10.1038/s41586-019-1724-z.
URL https://www.nature.com/articles/s41586-019-1724-z

[6] P. Henderson, J. Hu, J. Romoff, E. Brunskill, D. Jurafsky, J. Pineau, Towards the Systematic Reporting of the Energy and Carbon Footprints of Machine Learning, Journal of Machine Learning Research 21 (2020) 1–43.
URL http://jmlr.org/papers/v21/20-312.html.

[7] K. Arulkumaran, M. P. Deisenroth, M. Brundage, A. A. Bharath, Deep reinforcement learning: A brief survey, IEEE Signal Processing Magazine 34 (6) (2017) 26–38. doi:10.1109/MSP.2017.2743240.

[8] T. Aotani, T. Kobayashi, K. Sugimoto, Bottom-up multi-agent reinforcement learning by reward shaping for cooperative-competitive tasks, Applied Intelligence 2021 51:7 51 (7) (2021) 4434–4452. doi:10.1007/S10489-020-02034-2.
URL https://link.springer.com/article/10.1007/s10489-020-02034-2

[9] S. Carta, A. Corriga, A. Ferreira, A. S. Podda, D. R. Recupero, A multi-layer and multi-ensemble stock trader using deep learning and deep reinforcement learning, Applied Intelligence 2020 51:2 51 (2) (2020) 889–905. doi:10.1007/

S10489-020-01839-5.
URL https://link.springer.com/article/10.1007/s10489-020-01839-5

[10] G. Paludo Licks, J. Colleoni Couto, P. de Fátima Miehe, R. de Paris, D. Dubugras Ruiz, F. Meneguzzi, SmartIX: A database indexing agent based on reinforcement learning, Applied Intelligence 2020 50:8 50 (8) (2020) 2575–2588. doi:10.1007/S10489-020-01674-8.
URL https://link.springer.com/article/10.1007/s10489-020-01674-8

[11] G. Zou, J. Tang, L. Yilmaz, X. Kong, Online food ordering delivery strategies based on deep reinforcement learning, Applied Intelligence 2021 (2021) 1–13doi:10.1007/S10489-021-02750-3.
URL https://link.springer.com/article/10.1007/s10489-021-02750-3

[12] M. Kusy, R. Zajdel, Probabilistic neural network training procedure based on Q(0)-learning algorithm in medical data classification, Applied Intelligence 2014 41:3 41 (3) (2014) 837–854. doi:10.1007/S10489-014-0562-9.
URL https://link.springer.com/article/10.1007/s10489-014-0562-9

[13] M. Mahmud, M. S. Kaiser, A. Hussain, S. Vassanelli, Applications of Deep Learning and Reinforcement Learning to Biological Data, IEEE Transactions on Neural Networks and Learning Systems 29 (6) (2018) 2063–2079. doi:10.1109/TNNLS.2018.2790388.

[14] T. Kobayashi, Student-t policy in reinforcement learning to acquire global optimum of robot control, Applied Intelligence 2019 49:12 49 (12) (2019) 4335–4347. doi:10.1007/S10489-019-01510-8.
URL https://link.springer.com/article/10.1007/s10489-019-01510-8

[15] B. Kiumarsi, K. G. Vamvoudakis, H. Modares, F. L. Lewis, Optimal and Autonomous Control Using Reinforcement Learning: A Survey, IEEE Transactions on Neural Networks and Learning Systems 29 (6) (2018) 2042–2062. doi:10.1109/TNNLS.2017.2773458.

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

[17] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, D. Hassabis, Mastering the game of Go with deep neural networks and tree search, Nature 529 (7587) (2016) 484–489. arXiv:1610.00633, doi:10.1038/nature16961.

[18] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, T. Lillicrap, K. Simonyan, D. Hassabis, A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play., Science (New York, N.Y.) 362 (6419) (2018) 1140–1144. doi:10.1126/science.aar6404.
URL http://www.ncbi.nlm.nih.gov/pubmed/30523106

[19] D. Silver, S. Singh, D. Precup, R. S. Sutton, Reward is enough, Artificial Intelligence 299 (2021) 103535. doi:10.1016/J.ARTINT.2021.103535.

[20] P.-A. Andersen, M. Goodwin, O.-C. Granmo, Towards safe reinforcement-learning in industrial grid-warehousing, Information Sciences 537 (2020) 467–484. doi:https://doi.org/10.1016/j.ins.2020.06.010.
URL https://www.sciencedirect.com/science/article/pii/S0020025520305740

[21] Z. Ding, H. Dong, Challenges of Reinforcement Learning, Deep Reinforcement Learning: Fundamentals, Research and Applications (2020) 249–272doi:10.1007/978-981-15-4095-0_7.
URL https://link.springer.com/chapter/10.1007/978-981-15-4095-0{_}7

[22] S. Ontanon, The combinatorial multi-armed bandit problem and its application to real-time strategy games, in: Proceedings, The Ninth AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment, 2013, pp. 58–64.
URL http://www.aaai.org/ocs/index.php/AIIDE/AIIDE13/paper/viewPaper/7377

[23] P.-A. Andersen, M. Goodwin, O. Granmo, Deep RTS: A Game Environment for Deep Reinforcement Learning in Real-Time Strategy Games, in: 2018 IEEE Conference on Computational Intelligence and Games (CIG), 2018, pp. 1–8. doi:10.1109/CIG.2018.8490409.

[24] C. Berner, G. Brockman, B. Chan, V. Cheung, P. Debiak, C. Dennison, D. Farhi, Q. Fischer, S. Hashme, C. Hesse, R. Józefowicz, S. Gray, C. Olsson, J. Pachocki, M. Petrov, H. P. de Oliveira Pinto, J. Raiman, T. Salimans, J. Schlatter, J. Schneider, S. Sidor, I. Sutskever, J. Tang, F. Wolski, S. Zhang, Dota 2 with Large Scale Deep Reinforcement Learning, CoRR abs/1912.0 (2019).
URL http://arxiv.org/abs/1912.06680

[25] H. Sethy, A. Patel, V. Padmanabhan, Real Time Strategy Games: A Reinforcement Learning Approach, Procedia Computer Science 54 (2015) 257–264. doi:10.1016/J.PROCS.2015.06.030.

[26] S. Levine, C. Finn, T. Darrell, P. Abbeel, End-to-end training of deep visuomotor policies, Journal of Machine Learning Research 17 (1) (2016) 1334–1373.
URL http://www.jmlr.org/papers/volume17/15-522/15-522.pdf

[27] X. Hu, T. Liu, X. Qi, M. Barth, Reinforcement Learning for Hybrid and Plug-In Hybrid Electric Vehicle Energy Management: Recent Advances and Prospects, IEEE Industrial Electronics Magazine 13 (3) (2019) 16–25. doi:10.1109/MIE.2019.2913015.

[28] G. E. Monahan, State of the Art—A Survey of Partially Observable Markov Decision Processes: Theory, Models, and Algorithms, Management Science 28 (1) (1982) 1–16. doi:10.1287/MNSC.28.1.1.
URL https://pubsonline.informs.org/doi/abs/10.1287/mnsc.28.1.1

[29] M. L. Puterman, Chapter 8 Markov decision processes, Handbooks in Operations Research and Management Science 2 (C) (1990) 331–434. doi:10.1016/S0927-0507(05)80172-0.

[30] M. Hausknecht, P. Stone, Deep Recurrent Q-Learning for Partially Observable MDPs, in: AAAI Fall Symposium Series, 2015, pp. 1–9. arXiv:1507.06527.

[31] M. L. Littman, A. R. Cassandra, L. P. Kaelbling, Learning policies for partially observable environments: Scaling up, Machine Learning Proceedings 1995 (1995) 362–370doi:10.1016/B978-1-55860-377-6.50052-9.

[32] A. Rodriguez, R. Parr, D. Koller, Reinforcement Learning Using Approximate Belief States, in: S. Solla, T. Leen, K. Müller (Eds.), Advances in Neural Information Processing Systems, Vol. 12, MIT Press, 2000, pp. 1036–1042.
URL https://proceedings.neurips.cc/paper/1999/file/158fc2ddd52ec2cf54d3c161f2dd6517-Paper.pdf

[33] X. Xiang, S. Foo, Recent Advances in Deep Reinforcement Learning Applications for Solving Partially Observable Markov Decision Processes (POMDP) Problems: Part 1—Fundamentals and Applications in Games, Robotics and Natural Language Processing, Machine Learning and Knowledge Extraction 2021, Vol. 3, Pages 554-581 3 (3) (2021) 554–581. doi:10.3390/MAKE3030029.
URL https://www.mdpi.com/2504-4990/3/3/29/htmhttps://www.mdpi.com/2504-4990/3/3/29

[34] R. Bellman, On the Theory of Dynamic Programming, Proceedings of the National Academy of Sciences 38 (8) (1952) 716–719. doi:10.1073/PNAS.38.8.716.
URL https://www.pnas.org/content/38/8/716https://www.pnas.org/content/38/8/716.abstract

[35] L. P. Kaelbling, M. L. Littman, A. W. Moore, Reinforcement Learning: A Survey, Journal of Artificial Intelligence Research (apr 1996). arXiv:9605103, doi:10.1.1.68.466.
URL http://arxiv.org/abs/cs/9605103

[36] M. Van Otterlo, M. Wiering, Reinforcement learning and markov decision processes, Adaptation, Learning, and Optimization 12 (2012) 3–42. doi:10.1007/978-3-642-27645-3_1.

[37] T. M. Moerland, J. Broekens, C. M. Jonker, Model-based Reinforcement Learning: A Survey, arxiv preprint arXiv:2006.16712 (jun 2020). arXiv:2006.16712.

URL https://arxiv.org/abs/2006.16712

[38] P.-A. Andersen, M. Goodwin, O.-C. Granmo, Towards Model-Based Reinforcement Learning for Industry-Near Environments, in: M. Bramer, M. Petridis (Eds.), Artificial Intelligence XXXVI, Springer International Publishing, Cham, 2019, pp. 36–49. doi:10.1007/978-3-030-34885-4_3.

[39] R. S. Sutton, Integrated Architectures for Learning, Planning, and Reacting Based on Approximating Dynamic Programming, Machine Learning Proceedings 1990 (1990) 216–224doi:10.1016/B978-1-55860-141-3.50030-4.

[40] S. R. Eddy, What is a hidden Markov model?, Nature Biotechnology 2004 22:10 22 (10) (2004) 1315–1316. doi:10.1038/nbt1004-1315.
URL https://www.nature.com/articles/nbt1004-1315

[41] D. P. Kingma, M. Welling, Auto-Encoding Variational Bayes, Proceedings of the 2nd International Conference on Learning Representations (dec 2013). arXiv:1312.6114, doi:10.1051/0004-6361/201527329.
URL http://arxiv.org/abs/1312.6114

[42] R. E. Kalman, A new approach to linear filtering and prediction problems, Journal of Fluids Engineering, Transactions of the ASME 82 (1) (1960). doi:10.1115/1.3662552.

[43] S. Hochreiter, J. Schmidhuber, Long Short-Term Memory, Neural Computation 9 (8) (1997). doi:10.1162/neco.1997.9.8.1735.

[44] A. Van Den Oord, O. Vinyals, K. Kavukcuoglu, Neural discrete representation learning, in: Advances in Neural Information Processing Systems, Vol. 2017-Decem, 2017, pp. 1–11.

[45] J. Durbin, S. J. Koopman, Time Series Analysis by State Space Methods: Second Edition, Oxford Statistical Science Series (2012).

[46] C. J. C. H. Watkins, P. Dayan, Q-learning, Machine Learning 1992 8:3 8 (3) (1992) 279–292. doi:10.1007/BF00992698.
URL https://link.springer.com/article/10.1007/BF00992698

[47] T. Jaakkola, M. Jordan, S. Singh, On the Convergence of Stochastic Iterative Dynamic Programming Algorithms, Neural Computation 6 (6) (1994) 1185–1201. doi:10.1162/neco.1994.6.6.1185.

[48] R. S. Sutton, The Bitter Lesson (2019).
URL http://www.incompleteideas.net/IncIdeas/BitterLesson.html

[49] J. Schulman, P. Moritz, S. Levine, M. I. Jordan, P. Abbeel, High-dimensional continuous control using generalized advantage estimation, in: 4th International Conference on Learning Representations, ICLR 2016 - Conference Track Proceedings, 2016, pp. 1–14.

[50] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, O. Klimov, Proximal Policy Optimization Algorithms, arxiv preprint arXiv:1707.06347 (jul 2017). arXiv:1707.06347.
URL http://arxiv.org/abs/1707.06347

[51] J. Schulman, S. Levine, P. Abbeel, M. Jordan, P. Moritz, Trust Region Policy Optimization, in: F. Bach, D. Blei (Eds.), Proceedings of the 32nd International Conference on Machine Learning, Vol. 37 of Proceedings of Machine Learning Research, PMLR, Lille, France, 2015, pp. 1889–1897.
URL http://proceedings.mlr.press/v37/schulman15.html

[52] A. Kendall, Y. Gal, What uncertainties do we need in Bayesian deep learning for computer vision?, in: Advances in Neural Information Processing Systems, Vol. 2017-Decem, 2017, pp. 5580–5590.
URL http://papers.nips.cc/paper/7141-what-uncertainties-do-we-need-in-bayesian-deep-learning-for-computer-vision

[53] J. A. Bagnell, A. Y. Ng, J. G. Schneider, Solving Uncertain Markov Decision Processes, Carnegie Mellon Research Showcase (2001).

[54] M. Kearns, S. Singh, Near-Optimal Reinforcement Learning in Polynomial Time, Machine Learning 2002 49:2 49 (2) (2002) 209–232. doi:10.1023/A:1017984413808.

URL https://link.springer.com/article/10.1023/A:1017984413808

[55] J. Schmidhuber, Formal theory of creativity, fun, and intrinsic motivation (1990-2010), IEEE Transactions on Autonomous Mental Development 2 (3) (2010) 230–247. doi:10.1109/TAMD.2010.2056368.

[56] D. Pathak, P. Agrawal, A. A. Efros, T. Darrell, Curiosity-driven exploration by self-supervised prediction, in: Proc. 34th International Conference on Machine Learning, ICML'17, Vol. 70, JMLR.org, Sydney, NSW, Australia, 2017, pp. 2778–2787.
URL https://dl.acm.org/doi/10.5555/3305890.3305968

[57] L. L. Edith, C. Melanie, P. Doina, R. Bohdana, Risk-directed Exploration in Reinforcement Learning, IJCAI 2005 Workshop on Planning and Learning in A Priori Unknown or Dynamic Domains (2005).

[58] J. Yang, W. Qiu, A measure of risk and a decision-making model based on expected utility and entropy, in: European Journal of Operational Research, 2005, pp. 792–799. doi:10.1016/j.ejor.2004.01.031.

[59] A. Tijsma, M. Drugan, M. Wiering, Comparing exploration strategies for Q-learning in random stochastic mazes, 2016 IEEE Symposium Series on Computational Intelligence, SSCI 2016 (feb 2017). doi:10.1109/SSCI.2016.7849366.

[60] P. Geibel, F. Wysotzki, Risk-Sensitive Reinforcement Learning Applied to Control under Constraints, Journal of Artificial Intelligence Research 24 (2005) 81–108. doi:10.1613/jair.1666.

[61] O. Mihatsch, R. Neuneier, Risk-sensitive reinforcement learning, Machine learning 49 (2) (2002) 267–290.

[62] P. Campos, T. Langlois, Abalearn: Efficient self-play learning of the game abalone, INESC-ID, neural networks and signal processing group (2003).

[63] A. Gosavi, Reinforcement learning: A tutorial survey and recent advances, INFORMS Journal on Computing 21 (2) (2009) 178–192.

[64] J. Romoff, P. Henderson, A. Piche, V. Francois-Lavet, J. Pineau, Reward Estimation for Variance Reduction in Deep Reinforcement Learning, in: A. Billard, A. Dragan, J. Peters, J. Morimoto (Eds.), Proceedings of The 2nd Conference on Robot Learning, Vol. 87 of Proceedings of Machine Learning Research, PMLR, 2018, pp. 674–699.
URL https://proceedings.mlr.press/v87/romoff18a.html

[65] J. del R. Millán, Reinforcement learning of goal-directed obstacle-avoiding reaction strategies in an autonomous mobile robot, Robotics and Autonomous Systems 15 (4) (1995) 275–299. doi:10.1016/0921-8890(95)00021-7.

[66] P.-A. Andersen, M. Goodwin, O.-C. Granmo, CostNet: An End-to-End Framework for Goal-Directed Reinforcement Learning, Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics) 12498 LNAI (2020) 94–107. doi:10.1007/978-3-030-63799-6_7.
URL https://link.springer.com/chapter/10.1007/978-3-030-63799-6{_}7

[67] Y. Lu, Artificial intelligence: a survey on evolution, models, applications and future trends, https://doi.org/10.1080/23270012.2019.1570365 6 (1) (2019) 1–29. doi:10.1080/23270012.2019.1570365.
URL https://www.tandfonline.com/doi/abs/10.1080/23270012.2019.1570365

[68] F. Berkenkamp, M. Turchetta, A. P. Schoellig, A. Krause, Safe Model-based Reinforcement Learning with Stability Guarantees, in: I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, R. Garnett (Eds.), Advances in Neural Information Processing Systems 30, Curran Associates, Inc., Long Beach, CA, USA, 2017, pp. 908–918.
URL https://papers.nips.cc/paper/6692-safe-model-based-reinforcement-learning-with-stability-guarantees

[69] Y. Chow, M. Ghavamzadeh, L. Janson, M. Pavone, Risk-Constrained Reinforcement Learning with Percentile Risk Criteria, Journal of Machine Learning Research 18 (1) (2017) 6070–6120.
URL http://www.jmlr.org/papers/volume18/15-636/15-636.pdf

[70] R. Cheng, G. Orosz, R. M. Murray, J. W. Burdick, End-to-end safe reinforcement learning through barrier functions for safety-critical continuous control tasks, in: 33rd AAAI Conference on Artificial Intelligence, AAAI 2019, 31st Innovative

Applications of Artificial Intelligence Conference, IAAI 2019 and the 9th AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, 2019. doi:10.1609/aaai.v33i01.33013387.

[71] J. Yang, W. Qiu, A measure of risk and a decision-making model based on expected utility and entropy, European Journal of Operational Research 164 (3) (2005) 792–799. doi:10.1016/J.EJOR.2004.01.031.

[72] W. Saunders, G. Sastry, A. Stuhlmüller, O. Evans, Trial without Error: Towards Safe Reinforcement Learning via Human Intervention, in: Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems, AAMAS '18, International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 2018, pp. 2067–2069.

[73] M. Turchetta, A. Kolobov, S. Shah, A. Krause, A. Agarwal, Safe Reinforcement Learning via Curriculum Induction, in: H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, H. Lin (Eds.), Advances in Neural Information Processing Systems, Vol. 33, Curran Associates, Inc., 2020, pp. 12151–12162.
URL https://proceedings.neurips.cc/paper/2020/file/8df6a65941e4c9da40a4fb899de65c55-Paper.pdf

[74] F. Berkenkamp, A. Krause, A. P. Schoellig, Bayesian optimization with safety constraints: safe and automatic parameter tuning in robotics, Machine Learning 2021 (2021) 1–35doi:10.1007/S10994-021-06019-1.
URL https://link.springer.com/article/10.1007/s10994-021-06019-1

[75] P.-A. Andersen, M. Goodwin, O.-C. Granmo, Towards a Deep Reinforcement Learning Approach for Tower Line Wars, in: M. Bramer, M. Petridis (Eds.), Artificial Intelligence XXXIV, Springer International Publishing, Cham, 2017, pp. 101–114. doi:10.1007/978-3-319-71078-5_8.

[76] Y. Tian, Q. Gong, W. Shang, Y. Wu, C. L. Zitnick, ELF: An Extensive, Lightweight and Flexible Research Platform for Real-time Strategy Games, Advances in Neural Information Processing Systems (2017) 2656–2666arXiv:1707.01067.
URL http://arxiv.org/abs/1707.01067

[77] O. Vinyals, T. Ewalds, S. Bartunov, P. Georgiev, A. S. Vezhnevets, M. Yeo, A. Makhzani, H. Küttler, J. Agapiou, J. Schrittwieser, J. Quan, S. Gaffney, S. Petersen, K. Simonyan, T. Schaul, H. van Hasselt, D. Silver, T. Lillicrap, K. Calderone, P. Keet, A. Brunasso, D. Lawrence, A. Ekermo, J. Repp, R. Tsing, StarCraft II: A New Challenge for Reinforcement Learning, Proceedings, The Thirteenth AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (2017) 86–92arXiv:1708.04782, doi:https://deepmind.com/documents/110/sc2le.pdf.
URL http://arxiv.org/abs/1708.04782

[78] Z.-J. Pang, R.-Z. Liu, Z.-Y. Meng, Y. Zhang, Y. Yu, T. Lu, On Reinforcement Learning for Full-Length Game of StarCraft, Proceedings of the AAAI Conference on Artificial Intelligence 33 (01) (2019) 4691–4698. doi:10.1609/AAAI.V33I01.33014691.
URL https://ojs.aaai.org/index.php/AAAI/article/view/4394

[79] S. Ontanon, G. Synnaeve, A. Uriarte, F. Richoux, D. Churchill, M. Preuss, A survey of real-time strategy game AI research and competition in starcraft, IEEE Transactions on Computational Intelligence and AI in Games 5 (4) (2013) 293–311. doi:10.1109/TCIAIG.2013.2286295.
URL http://ieeexplore.ieee.org/document/6637024/

[80] N. A. Barriga, M. Stanescu, M. Buro, Game Tree Search Based on Non-Deterministic Action Scripts in Real-Time Strategy Games, IEEE Transactions on Computational Intelligence and AI in Games (2017) 1–1doi:10.1109/TCIAIG.2017.2717902.
URL http://ieeexplore.ieee.org/document/7954767/

[81] A. Shleyfman, A. Komenda, C. Domshlak, On combinatorial actions and CMABs with linear side information, in: Frontiers in Artificial Intelligence and Applications, Vol. 263, 2014, pp. 825–830. doi:10.3233/978-1-61499-419-0-825.

[82] M. Botvinick, S. Ritter, J. X. Wang, Z. Kurth-Nelson, C. Blundell, D. Hassabis, Reinforcement Learning, Fast and Slow., Trends in cognitive sciences 23 (5) (2019) 408–422. doi:10.1016/j.tics.2019.02.006.

URL http://www.ncbi.nlm.nih.gov/pubmed/31003893

[83] K. J. Roodbergen, I. F. A. Vis, A survey of literature on automated storage and retrieval systems, European Journal of Operational Research (2009). doi:10.1016/j.ejor.2008.01.038.

[84] M. Fraccaro, Deep latent variable models for sequential data, Ph.D. thesis, Technical University of Denmark (2018). URL https://orbit.dtu.dk/en/publications/deep-latent-variable-models-for-sequential-data

[85] A. Razavi, A. van den Oord, O. Vinyals, Generating Diverse High-Fidelity Images with VQ-VAE-2, in: H. Wallach, H. Larochelle, A. Beygelzimer, F. Alché-Buc, E. Fox, R. Garnett (Eds.), Advances in Neural Information Processing Systems 32, Curran Associates, Inc., Vancouver, BC, Canada, 2019, pp. 14837–14847. URL http://papers.nips.cc/paper/9625-generating-diverse-high-fidelity-images-with-vq-vae-2

[86] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, Proceedings, IEEE 86 (11) (1998) 2278–2323. arXiv:1102.0183, doi:10.1109/5.726791. URL http://ieeexplore.ieee.org/document/726791/

[87] P.-A. Andersen, M. Goodwin, O.-C. Granmo, The Dreaming Variational Autoencoder for Reinforcement Learning Environments, in: Max Bramer, M. Petridis (Eds.), Artificial Intelligence XXXV, xxxv Edition, Vol. 11311, Springer, Cham, 2018, pp. 143–155. doi:10.1007/978-3-030-04191-5_11.

[88] A. Doerr, C. Daniel, M. Schiegg, N.-T. Duy, S. Schaal, M. Toussaint, T. Sebastian, Probabilistic Recurrent State-Space Models, in: J. Dy, A. Krause (Eds.), Proceedings of the 35th International Conference on Machine Learning, Vol. 80 of Proceedings of Machine Learning Research, PMLR, 2018, pp. 1280–1289. URL http://proceedings.mlr.press/v80/doerr18a.html

[89] C. Zhang, J. Butepage, H. Kjellstrom, S. Mandt, Advances in Variational Inference, IEEE Transactions on Pattern Analysis and Machine Intelligence 41 (8) (2019) 2008–2026. doi:10.1109/TPAMI.2018.2889774.

[90] M. I. Jordan, Z. Ghahramani, T. S. Jaakkola, L. K. Saul, Introduction to variational methods for graphical models, Machine Learning 37 (2) (1999) 183–233. doi:10.1023/A:1007665907178. URL https://link.springer.com/article/10.1023/A:1007665907178

[91] R. S. Sutton, D. Precup, S. Singh, Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning, Artificial Intelligence 112 (1-2) (1999) 181–211.

[92] S. Ozair, Y. Li, A. Razavi, I. Antonoglou, A. van den Oord, O. Vinyals, Vector Quantized Models for Planning, in: Proc.. 38th International Conference on Machine Learning, PMLR 139, 2021, 2021, pp. 1–15. arXiv:2106.04615. URL http://arxiv.org/abs/2106.04615

[93] P. Seetharaman, G. Wichern, B. Pardo, J. L. Roux, Autoclip: Adaptive gradient clipping for source separation networks, in: IEEE International Workshop on Machine Learning for Signal Processing, MLSP, Vol. 2020-Septe, IEEE Computer Society, 2020, pp. 1–6. arXiv:2007.14469, doi:10.1109/MLSP49062.2020.9231926.

[94] I. Loshchilov, F. Hutter, Decoupled Weight Decay Regularization, in: Proc. 7th International Conference on Learning Representations, ICLR'19, 2019, pp. 1–19. URL https://openreview.net/forum?id=Bkg6RiCqY7

[95] P. Izmailov, D. Podoprikhin, T. Garipov, D. Vetrov, A. G. Wilson, Averaging Weights Leads to Wider Optima and Better Generalization, in: A. G. R. Silva, A. Globerson (Eds.), 34th Conference on Uncertainty in Artificial Intelligence 2018, Association For Uncertainty in Artificial Intelligence, 2018, pp. 876–885. arXiv:1803.05407. URL http://arxiv.org/abs/1803.05407

[96] P.-A. Andersen, M. Goodwin, O. Granmo, Increasing sample efficiency in deep reinforcement learning using generative environment modelling, Expert Systems (mar 2020). doi:10.1111/exsy.12537. URL https://onlinelibrary.wiley.com/doi/abs/10.1111/exsy.12537

[97] S. Huang, S. Ontanon, C. Bamford, L. Grela, Gym-MicroRTS: Toward Affordable Full Game Real-time Strategy Games

Research with Deep Reinforcement Learning, in: Proc. 3rd IEEE Conference on Games, 2021, p. 19. arXiv:2105.13807. URL https://arxiv.org/abs/2105.13807v3

[98] H. Hu, Q. Wang, Implementation on benchmark of SC2LE environment with advantage actor - Critic method, 2020 International Conference on Unmanned Aircraft Systems, ICUAS 2020 (2020) 362–366doi:10.1109/ICUAS48674.2020.9214032.

[99] Z. Zha, B. Wang, X. Tang, Evaluate, explain, and explore the state more exactly: an improved Actor-Critic algorithm for complex environment, Neural Computing and Applications 2021 (2021) 1–12doi:10.1007/S00521-020-05663-3. URL https://link.springer.com/article/10.1007/s00521-020-05663-3

[100] M. Janner, J. Fu, M. Zhang, S. Levine, When to Trust Your Model: Model-Based Policy Optimization, in: H. Wallach, H. Larochelle, A. Beygelzimer, F. Alché-Buc, E. Fox, R. Garnett (Eds.), Proc. 33rd Conference on Neural Information Processing Systems (NeurIPS), Curran Associates, Inc., Vancouver, BC, Canada, 2019, pp. 12519–12530. URL http://papers.nips.cc/paper/9416-when-to-trust-your-model-model-based-policy-optimization.pdf

[101] D. Hafner, T. Lillicrap, J. Ba, M. Norouzi, Dream to Control: Learning Behaviors by Latent Imagination, in: Proc. 8th International Conference on Learning Representations, ICLR'20, 2020, pp. 1–26. URL https://openreview.net/forum?id=S1lOTC4tDS

[102] A. Razavi, A. van den Oord, B. Poole, O. Vinyals, Preventing Posterior Collapse with delta-VAEs, in: Proc. 7th International Conference on Learning Representations, ICLR'19, 2019, pp. 1–24. URL https://openreview.net/forum?id=BJe0Gn0cY7

Appendix A. Micro RTS Results

Table A.11: MicroRTS-basesWorkers8x8A results.

basesWorkers8x8A						
Agorithm	RandomBiasedAI	NaiveMCTS	Tiamat	Droplet	UTS_Imass	Win Ratio
A2C	0.99 ±0.01	0.88 ±0.06	0.46 ±0.06	0.65 ±0.27	0.46 ±0.19	0.69
DQN	0.99 ±0.01	0.86 ±0.06	0.67 ±0.07	0.56 ±0.11	0.64 ±0.24	0.74
RAINBOW	1	0.89 ±0.07	0.76 ±0.16	0.62 ±0.27	0.67 ±0.09	0.79
IMPALA	0.99 ±0.01	0.79 ±0.08	0.51 ±0.27	0.73 ±0.2	0.42 ±0.4	0.69
PPO	1	0.92 ±0.06	0.66 ±0.3	0.83 ±0.12	0.75 ±0.25	0.83
DVAE	0.96 ±0.03	0.87 ±0.03	0.42 ±0.07	0.73 ±0.03	0.56 ±0.19	0.71
S-ORACLE	0.98 ±0.02	0.87 ±0.01	0.63 ±0.32	0.75 ±0.08	0.57 ±0.41	0.76

Table A.12: MicroRTS-basesWorkers16x16A results.

basesWorkers16x16A						
Agorithm	RandomBiasedAI	NaiveMCTS	Tiamat	Droplet	UTS_Imass	Win Ratio
A2C	0.93 ±0.05	0.75 ±0.12	0.54 ±0.36	0.47 ±0.02	0.61 ±0.1	0.66
DQN	0.91 ±0.01	0.72 ±0.21	0.57 ±0.3	0.75 ±0.21	0.52 ±0.04	0.69
RAINBOW	0.9 ±0.05	0.63 ±0.15	0.73 ±0.25	0.55 ±0.31	0.75 ±0.15	0.71
IMPALA	0.85 ±0.06	0.64 ±0.16	0.66 ±0.07	0.76 ±0.19	0.57 ±0.26	0.7
PPO	0.87 ±0.03	0.7 ±0.24	0.46 ±0.36	0.86 ±0.09	0.57 ±0.19	0.69
DVAE	0.9 ±0.1	0.7 ±0.04	0.66 ±0.22	0.67 ±0.15	0.77 ±0.18	0.74
S-ORACLE	0.94 ±0.06	0.75 ±0.23	0.61 ±0.03	0.66 ±0.18	0.75 ±0.21	0.74

Table A.13: MicroRTS-BWDistantResources32x32 results.

BWDistantResources32x32						
Agorithm	RandomBiasedAI	NaiveMCTS	Tiamat	Droplet	UTS_Imass	Win Ratio
A2C	0.87 ±0.03	0.67 ±0.12	0.86 ±0.13	0.66 ±0.04	0.83 ±0.12	0.78
DQN	0.84 ±0.08	0.87 ±0.12	0.77 ±0.04	0.86 ±0.06	0.82 ±0.12	0.83
RAINBOW	0.81 ±0.16	0.99 ±0.01	0.76 ±0.05	0.89 ±0.08	0.86 ±0.13	0.86
IMPALA	0.87 ±0.11	0.94 ±0.05	0.97 ±0.02	0.95 ±0.04	0.91 ±0.07	0.93
PPO	0.9 ±0.08	0.97 ±0.02	0.93 ±0.01	0.91 ±0.01	0.81 ±0.02	0.9
DVAE	0.9 ±0.03	0.95 ±0.01	0.78 ±0.19	0.97 ±0.02	0.78 ±0.03	0.88
S-ORACLE	0.91 ±0.05	0.94 ±0.06	0.97 ±0.02	0.91 ±0.02	0.83 ±0.11	0.91

Table A.14: MicroRTS-DoubleGame24x24 results.

DoubleGame24x24						
Agorithm	RandomBiasedAI	NaiveMCTS	Tiamat	Droplet	UTS_Imass	Win Ratio
A2C	0.78 ±0.01	0.97 ±0.03	0.89 ±0.09	0.89 ±0.05	0.95 ±0.05	0.9
DQN	0.81 ±0.13	0.91 ±0.04	0.94 ±0.02	0.89 ±0.03	0.32 ±0.67	0.77
RAINBOW	0.8 ±0.13	0.95 ±0.02	0.84 ±0.14	0.69 ±0.25	0.64 ±0.16	0.78
IMPALA	0.84 ±0.16	0.92 ±0.02	0.88 ±0.1	0.96 ±0.02	0.85 ±0.11	0.89
PPO	0.85 ±0.01	0.99 ±0.01	0.89 ±0.03	0.68 ±0.04	0.85 ±0.1	0.85
DVAE	0.75 ±0.04	0.96 ±0.03	0.93 ±0.07	0.96 ±0.03	0.95 ±0.01	0.91
S-ORACLE	0.73 ±0.27	0.92 ±0.08	0.95 ±0.05	0.91 ±0.03	0.91 ±0.01	0.88

Table A.15: MicroRTS-FourBasesWorkers8x8 results.

FourBasesWorkers8x8						
Agorithm	RandomBiasedAI	NaiveMCTS	Tiamat	Droplet	UTS_Imass	Win Ratio
A2C	1.0 ±0.0	0.91 ±0.08	0.95 ±0.05	0.84 ±0.14	0.81 ±0.17	0.9
DQN	1.0 ±0.0	0.89 ±0.03	0.78 ±0.03	0.64 ±0.12	0.71 ±0.08	0.8
RAINBOW	1.0 ±0.0	0.96 ±0.02	0.76 ±0.12	0.75 ±0.23	0.64 ±0.09	0.82
IMPALA	1.0 ±0.0	0.67 ±0.09	0.75 ±0.25	0.77 ±0.11	0.8 ±0.06	0.8
PPO	1.0 ±0.0	0.87 ±0.02	0.81 ±0.1	0.75 ±0.08	0.81 ±0.01	0.85
DVAE	1.0 ±0.0	0.75 ±0.12	0.75 ±0.03	0.96 ±0.03	0.64 ±0.36	0.82
S-ORACLE	1.0 ±0.0	0.87 ±0.02	0.82 ±0.13	0.67 ±0.24	0.78 ±0.02	0.83

Table A.16: MicroRTS-NoWhereToRun9x8 results.

NoWhereToRun9x8						
Agorithm	RandomBiasedAI	NaiveMCTS	Tiamat	Droplet	UTS_Imass	Win Ratio
A2C	1.0 ±0.0	0.86 ±0.13	0.75 ±0.14	0.85 ±0.03	0.89 ±0.02	0.87
DQN	1.0 ±0.0	0.88 ±0.04	0.66 ±0.1	0.81 ±0.19	0.86 ±0.02	0.84
RAINBOW	0.98 ±0.01	0.82 ±0.17	0.89 ±0.05	0.86 ±0.02	0.84 ±0.16	0.88
IMPALA	1.0 ±0.0	0.93 ±0.02	0.96 ±0.02	0.87 ±0.02	0.88 ±0.01	0.93
PPO	0.99 ±0.01	0.96 ±0.02	0.88 ±0.07	0.84 ±0.11	0.82 ±0.14	0.9
DVAE	0.99 ±0.01	0.91 ±0.04	0.75 ±0.23	0.83 ±0.15	0.88 ±0.07	0.87
S-ORACLE	1.0 ±0.0	0.91 ±0.02	0.89 ±0.03	0.88 ±0.12	0.89 ±0.05	0.91

Table A.17: MicroRTS-TwoBasesBarracks16x16 results.

TwoBasesBarracks16x16						
Agorithm	RandomBiasedAI	NaiveMCTS	Tiamat	Droplet	UTS_Imass	Win Ratio
A2C	0.97 ±0.01	0.91 ±0.04	0.97 ±0.01	0.89 ±0.07	0.89 ±0.04	0.93
DQN	0.96 ±0.02	0.9 ±0.02	0.89 ±0.1	0.89 ±0.1	0.78 ±0.03	0.88
RAINBOW	0.96 ±0.02	0.97 ±0.03	0.88 ±0.05	0.83 ±0.1	0.8 ±0.18	0.89
IMPALA	0.96 ±0.03	0.98 ±0.02	0.92 ±0.04	0.88 ±0.12	0.92 ±0.06	0.93
PPO	1.0 ±0.0	0.99 ±0.01	0.98 ±0.01	0.85 ±0.1	0.97 ±0.02	0.96
DVAE	0.96 ±0.01	0.89 ±0.03	0.85 ±0.02	0.99 ±0.01	0.86 ±0.12	0.91
S-ORACLE	0.97 ±0.01	0.9 ±0.04	0.82 ±0.06	0.9 ±0.09	0.88 ±0.04	0.89

Appendix B. Hyperparameters

Table B.18: Set of tunable parameters in S-ORACLE. In addition to this incomplete list, the algorithm has options for controlling model complexity such as neuron counts and number of layers.

Hyperparameter	**Values**	**Selected**	**Comment**
Batch Size	$\mathbb{Z}^+$	48	Number of sequence batches
Sequence Size	$\mathbb{Z}^+$	48	Number of frames in a sequence
Buffer Size	$\mathbb{Z}^+$	9 000	Replay buffer
Reward Scaling	$\mathbb{R}$	1.0	Scaling of the reward objective
Cost Scaling	$\mathbb{R}$	1.0	Scaling of the cost objective
VQ Scaling	$\mathbb{R}$	0.1	Scaling of the VQ objective
KL Scaling	$\mathbb{R}$	1.0	Scaling of the KL objective (KL-β)
KL Minimum Nats	$\mathbb{R}$	3.0	Minimal information loss
Optimizer		AdamW	AdamW improves generalization, see [94]
Gradient Clipping	$\mathbb{R}$	100.0 $\pm$	Clip gradients to increase learning stability
Adaptive GC	$\mathbb{B}$	1	Based on the history of gradient norms[93]
Learning Rate	$\mathbb{R}$	0.0001	Low Learning rate to improve stability.
Latent Leaps	$\mathbb{Z}^+$	30	Number of leaps into future states.
Dynamics Model RNN		LSTM	
Activation Functions		ELU	
Enc/Dec Neurons	$\mathbb{Z}^+$	1024	
Stochastic Reward	$\mathbb{B}$	1	Sample rewards under Gaussian assumptions
Stochastic Costs	$\mathbb{B}$	1	Sample costs under Gaussian assumptions
Discount Factor γ	$\mathbb{R}$	0.96	Discount factor used in value-based algorithms
Risk-Entropy Weight w (Eq 20)	$\mathbb{R}$	0.6	Risk-Directed Exploration entropy weight
Risk-Utility Function ρ(Eq 22)	$\mathbb{R}$	0.75	Weight of risk in utility function.
Risk function weight β (Eq 23)	$\mathbb{R}$	0.40	Weight of the risk functipn ω.
VQ-VAE weight β_{vq}	$\mathbb{R}$	0.05	Weight of VQ-VAE encoding updates.