Structured Multi-Agent World Models

New Jun Jie, Wu Yujin
School of Computing, National University of Singapore

Abstract

In multi-agent reinforcement learning, the difficulty of generalising to diverse strategies and adapting to non-stationary behaviour remains a challenge. Inspired by the effectiveness of model-based reinforcement learning, we present Structured Multi-Agent World Models (SMAWM), a world model that encompasses other agents in a compositional structure, to provide a strong inductive bias for generalising to novel interactions among multiple agents in the environment. We show that reinforcement learning with the agent-factored state representation outperforms that with a purely connectionist world model despite using much fewer parameters. We further show that SMAWM learns an effective representation that is capable of much higher accuracy in forward prediction for planning, and propose future extensions that can likely scale SMAWM to environments of higher complexity.

1 Introduction

Model-based reinforcement learning (MBRL) [9; 25] has proven promising in recent research, significantly outperforming their model-free counterparts in data efficiency, generalisation, exploration, counterfactual reasoning and performance in many tasks and domains [17; 18; 21; 30].

However, there remains challenges in MBRL for the multi-agent setting. Many real-world problems are multi-agent in nature, such as autonomous vehicles, and it remains difficult for reinforcement learning (RL) methods to generalise to novel observations of multi-agent interactions. Most MBRL methods assume the single-agent setting with independent learning, modeling interactions with other agents in the environment as noise, reducing the multi-agent learning problem to a single-agent one [4]. Independent learners may fail either due to the stochasticity of the environment or if the policies of other agents are non-stationary [4; 27]. Further, each agent has to consider other agents' actions, resulting in a large joint action space that grows exponentially with the number of agents, constraining the scalability of multi-agent reinforcement learning (MARL) methods. In contrast, opponent modeling allows predicting future actions of other agents through observing their actions, generalising to novel interactions more effectively [4].

Recent advances in graph-structured representations have significantly improved the predictive accuracy of models by explicitly taking into account the structured nature of tasks [5; 23]. The graph neural network with its compositional capabilities is thus a natural candidate for the multi-agent learning problem, where the state of an environment can be factored into individual agents and their interactions.

In this paper, we present Structured Multi-Agent World Models (SMAWM), a structured formulation of agent-factored state representations using graph neural networks to model environment dynamics and agent interactions for MBRL in the multi-agent mixed cooperative-competitive setting. SMAWM encompasses other agents in a compositional structure, modeling each individual agent's properties, and interactions and relationships among agents. We show that reinforcement learning with the agent-factored state representation in a multi-agent setting outperforms that with a purely connectionist world model despite using much fewer parameters in multiple settings of interest, and show that SMAWM is capable of much higher accuracy in forward prediction for planning.

2 Related Work

World Models World Models [8] learn latent dynamics using a variational autoencoder and a mixture density recurrent neural network to capture spatial and temporal information of states across time. The learnt model is then used to process raw images into the spatio-temporal latent representation for agent learning. PlaNet [10] learns spatial and temporal representations jointly and uses the learnt latent representations for online planning. Dreamer [9] learns latent dynamics and efficiently trains an agent purely within latent imagination. Our work most closely resembles World Models, but we employ a reinforcement learning algorithm as the agent, and study it in a multi-agent mixed cooperative-competitive setting.

Graph Neural Networks NPE [5] learns to factorize a state into object-based representations and object dynamics into pairwise interactions. Graph Networks [23] learns object- and relation-centric representations for planning and control. C-SWM [14] employs a contrastive learning approach to learn a set of object representations to represent states from an environment with compositional structure. Our work is inspired by C-SWM, but in contrast, we do not employ a contrastive loss for training the world model, and adapt the structured world model to model multi-agent interactions and dynamics for MBRL.

Multi-Agent Reinforcement Learning DRON [11] jointly learns a policy and the behavior of opponents using a deep Q-network. SOM [22] learns a policy and uses it model its belief of other agents' policy states. VAIN [12] learns a multi-agent attentional neural network for predictive modeling of local interactions. Our proposed method is much simpler in that it does not model the actions of other agents, but can be easily extended to do so.

3 Preliminaries

3.1 World Models

The world model, also known as an environment model or dynamics model, offers an explicit way to represent an agent's knowledge about the world in a parametric model that can make predictions about the future [9]. The world model learns to represent the state in a compressed yet informative abstract latent representation, and learns the state transition dynamics of the environment. The world model can be utilised in learning where synthetic trajectories may be sampled from the world model to reduce reliance on real-world data and improve data efficiency, or in planning where future trajectories are predicted ahead to plan for the best action to take in a given state [1; 19; 20].

World models commonly consist of 3 components:

$$\begin{aligned}&\text{Representation Model: } p_\theta(s_t|o_t)\\&\text{Transition Model: } p_\theta(s_{t+1}|s_t, a_t)\\&\text{Reward Model: } p_\theta(r_t|s_t)\end{aligned}$$

where p_θ can be neural networks parameterized with θ. o_t, s_t, a_t and r_t are observations, learnt states, actions and rewards at time t. In this work, we do not learn a reward model as we only utilise the representation model for feature extraction. In a future extension, we intend to learn an opponent model $p_\theta(a_{2:k}|s_t)$ where $a_{2:k}$ are the predicted actions of all $k-1$ agents other than the agent itself.

3.2 Graph Neural Networks

The graph neural network (GNN) is a class of neural networks that places an inductive bias on the structural relations of the input [3; 13]. A GNN assumes a data representation as multiple nodes $\{d_i\}_{i=1}^n D \in R^{d\times n}$ and an adjacency representation of edges $A_G \in [0,1]^{d\times d}$ in a graph G [31]. We use GNNs to implement the transition model, allowing modeling of pairwise interactions between agent states. The transition function takes as input a tuple of agent-factored state representations $s_t = (s_t^1, ..., s_t^K)$ and actions $a_t = (a_t^1, ..., a_t^K)$ at a particular time step:

$$\Delta s_t = T(s_t, a_t) = GNN(\{(s_t^k, a_t^k)\})$$

The transition model $T(s_t, a_t)$ is a GNN that predicts updates $\Delta s_t = (\Delta s_t^1, ..., \Delta s_t^K)$. The agent-factored state representation for the next time step are obtained via $s_{t+1} = (s_t^1 + \Delta s_t^1, ..., s_t^K + \Delta s_t^K$. The GNN consists of node update functions and edge update functions f_{node} and f_{edge} with shared parameters across all nodes and edges, implemented as MLPs with the following message passing updates:

$$e_t^{(i,j)} = f_{edge}([s_t^i, s_t^j])$$
$$\Delta s_t^j = f_{node}([s_t^j, a_t^j, \sum_{i \neq j} e_t^{(i,j)}])$$

where $e_t^{(i,j)}$ is an intermediate representation of the edge between nodes i and j, corresponding to a single round of node-to-edge and edge-to-node message passing [14].

3.3 Multi-Agent Reinforcement Learning

Multi-agent reinforcement learning addresses sequential decision making problems involving more than one agent, a generalization of the Markov decision process (MDP) to multiple agents as a Markov game, defined by a tuple $(N, S, \{U^i\}, P, \{R^i\}, \gamma)$ where N represents the set of interacting agents, S represents the state of all agents, U represents the action space of all agents, $P(S_{t+1}|S_t, U_t)$ denotes the environment dynamics, R_i represents the the individual reward functions and γ the discount factor of each agent [7]. Each agent optimizes its own value function, where π^i is the policy of each agent:

$$V^i_{\pi^i, \pi^{-i}} = E[\sum_{t \geq 0} \gamma^t R^i(s_t, a_t, s_{t+1}) | a_t^i \sim \pi(\cdot | s_t), s_0 = s]$$

In the multi-agent setting, the effectiveness of an agent's policy depends on that of other agents, especially when the joint policy of all k agents Π^k is non-stationary. Opponent modeling is a common approach to learn the joint policy Π^k or joint actions A^k of other agents [11].

4 Structured Multi-Agent World Models

4.1 Architecture

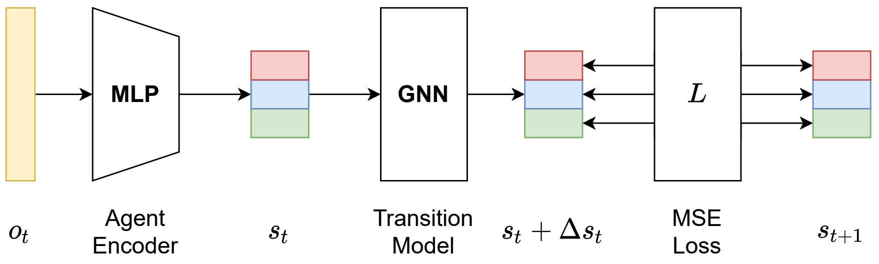


Figure 1: SMAWM consists of 3 components: a MLP-based agent encoder, a GNN-based transition model and an agent-factorized mean squared error (MSE) loss that jointly trains both the agent encoder and transition model.

Structured Multi-Agent World Models (SMAWM) is a world model inspired by C-SWM [14], but in contrast, consists of a MLP-based agent encoder, a MLP-based opponent model and a GNN-based transition model. Our goal is model the compositional nature of multiple agents in an environment and learn an agent-factored state space $s = s_1, ..., s_k$, where k is the number of agents.

The agent-factored state space serves as a strong inductive bias for better generalization to novel interactions, facilitating planning and decision making by the agent. The opponent model predicts the actions $a_{2:k}$ that would be taken by all other $k - 1$ agents. The transition model predicts the next state s_{t+1} given predicted agents' actions $\hat{a}_2, ..., \hat{a}_k$ and the agent's action a_1. The agent would use the learnt state representation model as a feature extractor, as in World Models (WM) [8].

Taking the off-policy setting, a dataset of offline experience $B = \{(o_t, a_t, o_{t+1})\}_{t=1}^{T}$ contains T tuples of observations $o_t \in O$, actions $a_t \in A$, and next observations $o_{t+1} \in O$ reached after taking action a_t in o_t. The goal is to learn abstract latent state representations $s_t \in S$ of environment observations $o_t \in O$ that retains information necessary for forward prediction of the next latent state representation s_{t+1}. The trained world model is then used for state representation to train the reinforcement learning agent.

4.2 Model and Hyperparameters

Agent-Factored Autoencoder The agent encoder is a MLP with two hidden layers of 10 units, and each followed by LayerNorm [2] and ReLU activation. The output of the final output layer is 5-dimensional for 3 agents, in total 15-dimensional. The agent decoder has the same architecture and number of hidden units as the agent encoder model, i.e., two hidden layers of 10 units each. The output of the agent decoder is the 10-dimensional observation.

Transition Model Both the node and edge model in the GNN-based transition model are MLPs with the same architecture and number of hidden units as the agent encoder model, i.e., two hidden layers of 10 units each, LayerNorm and ReLU activations. The number of parameters in the agent encoder and transition model combined is 1590, intentionally smaller than the baseline model for comparison. We do not predict future rewards or actions taken by other agents, which we leave for future extensions towards a similar direction as Dreamer [9].

Loss Function Instead of applying the contrastive loss as in C-SWM [14], we opt for a simpler mean squared error (MSE) loss for a fair comparison with a purely connectionist WM baseline. The agent encoder and transition model are jointly trained with a loss summed by the MSE between actual and reconstructed observations $L(o_t, \hat{o}_t)$ and the MSE between actual and predicted observations in the next time step $L(o_{t+1}, \hat{o}_{t+1})$.

5 Experiments

5.1 Multi-Agent Particle Environment

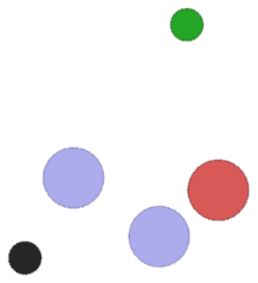


Figure 2: MPE: Simple Adversary is a mixed cooperative-competitive environment. Cooperative agents (blue) must approach the goal landmark (green) without allowing the adversarial agent (red) to infer and follow suit, possibly misleading it towards the non-goal landmark (black).

The Multi-Agent Particle Environment [15] is a set of environments where multiple particle agents can interact in cooperative and competitive settings. Simple Adversary is a task that is simple yet reflect interesting mixed cooperative-competitive interaction dynamics. Simple Adversary consists of two cooperative agents π_0 and π_1, one adversarial agent π_2, one goal and one non-goal landmark.

Reward Function Cooperative agents are rewarded based on the closeness of the closest one is to the goal, and negatively penalised by the adversary's closeness to that goal. The adversarial agent is rewarded based its closeness to the goal, but it cannot differentiate between goal and non-goal landmarks, and so it will try to infer the goal landmark from the behaviour of cooperative agents. The cooperative agents must cooperate to mislead the adversary away from the goal. The corresponding reward functions are thus:

$$R_{Cooperative} = -min(d(Goal, \pi_0), d(Goal, \pi_1)) + d(Goal, \pi_2)$$
$$R_{Adversarial} = -d(Goal, \pi_2)$$

where d is the Euclidean distance between entities, $R_{Cooperative} \in (-\infty, \infty)$ and $R_{Adversarial} \in [0, \infty)$, since the environment is unbounded.

Observation Space The observation space for a cooperative agent C_i contains the *relative* x,y-coordinates of entities from itself:

$$Observation_{C_i} : (\Delta x_G, \Delta y_G, \Delta x_{L1}, \Delta y_{L1}, \Delta x_{L2}, \Delta y_{L2}, \Delta x_A, \Delta y_A, \Delta x_{C_{j\neq i}}, \Delta y_{C_{j\neq i}})$$

where G denotes the goal landmark, $L1$ and $L2$ denotes arbitrary landmarks where one of it is randomly assigned as the goal, A denotes the adversarial agent, and C denotes the other cooperative agent.

Action Space $\{0, 1, 2, 3, 4\}$, where 0, 1, 2, 3, 4 corresponds to doing nothing, moving left, moving right, moving up and moving down respectively.

5.2 Random Actions Setting

To generate a dataset of offline experience for training the world model, a uniformly random policy where all actions are taken with an equal probability is used for all agents in the Simple Adversary and Simple Tag environments. A dataset of 1000 episodes with each episode having 100 time steps is generated to create the Random Actions dataset to train the world model.

5.3 Spurious Correlations Setting

In reality, datasets collected may contain spurious correlations between features. For example, a self-driving dataset may incidentally contain trajectories where a red and a blue car frequently driving together to the same destination, but that does not mean that any red car must necessarily take the same route as a blue car in the real world. To investigate the benefit of an agent-factored state representation, a Spurious Correlations dataset is generated by having one cooperative agent copy the same action as the other cooperative agent with 95% probability.

5.4 Expert Demonstrations Setting

The Random Actions and Spurious Correlations settings reflect either convenient or targeted contexts to compare SMAWM against WM. While the Random Actions setting is convenient as a baseline, it is an unrealistic dataset of offline experience. Random actions are only taken possibly at the start of agent training if the agent takes a random exploration approach, and in most realistic applications, random exploration is highly inefficient or unsafe. The Spurious Correlations setting is artificially constructed, presenting a very specific problem whose significance may not be much in more realistic contexts. Therefore, we construct an Expert Demonstrations dataset, where agents employ simple expert policies, as described in Section 5.5. The Expert Demonstrations setting studies a more realistic context, common in inverse reinforcement learning settings.

5.5 Benchmarking Environment

Simple expert cooperative and adversarial policies are constructed for a benchmarking environment used for evaluation. A simple cooperative policy is for the agent to always go to an assigned non-goal landmark, with a small ϵ probability of taking a random action. A simple adversarial policy is for the agent to go to the landmark closest to any agent with a small ϵ probability of taking a random action. The expert adversarial policy assumes that the goal landmark must be the one that is closest to an agent.

Algorithm 1 Cooperative Policy

```
if $random() \leq \epsilon$ then
    $action \leftarrow sample(Actions)$
else
    $action \leftarrow follow(Assigned)$
end if
```

Algorithm 2 Adversarial Policy

```
if $random() \leq \epsilon$ then
    $action \leftarrow sample(Actions)$
else
    $action \leftarrow follow(Closest)$
end if
```

The benchmarking environment assumes the expert cooperative and adversarial policies for one cooperative and one adversarial agent. The value of ϵ determines the effectiveness of the expert policies, and the difficulty of the task can be controlled by adjusting ϵ. We set $\epsilon = 0.2$ for both cooperative and adversarial policies to introduce some stochasticity yet try to minimise the variance in final performance. The agent to be trained takes the role of the cooperative agent assigned to the goal landmark and the Proximal Policy Optimization algorithm [26] is arbitrarily selected as the reinforcement learning algorithm of the agent, although any other algorithm could be used, e.g. A2C [16].

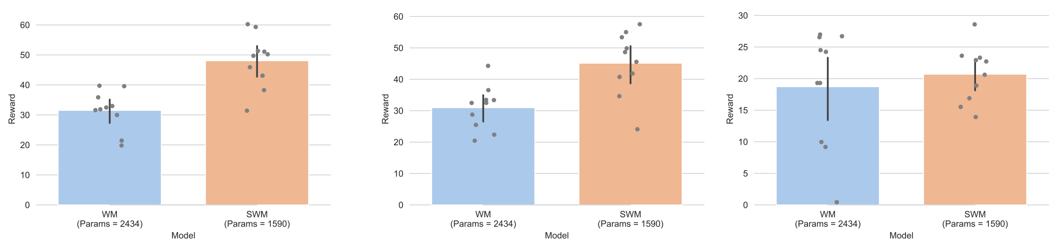
Each experiment procedure consists of training the world model, training the reinforcement learning agent on features extracted by the world model, before evaluating on the benchmark environment. Each experiment configuration is run across 10 random seeds to account for variance in results.

5.6 Baseline World Model

The baseline world model (WM) used for comparison [8] is purely connectionist. The architecture of the autoencoder in WM has the same hyperparameters (i.e. the same number of layers and units per layer) as the SMAWM, but it uses a variational autoencoder in contrast to the SMAWM that uses a simple autoencoder. The transition model in WM is implemented as a mixture density recurrent neural network that predicts the next state s_{t+1}, while the transition model of the SMAWM is implemented as a graph neural network that predicts the agent-factored change Δs_t to be applied to the current state s_t to obtain a prediction of the next state $s_{t+1} = s_t + \Delta s_t$. The WM serves as a purely connectionist baseline world model to contrast the difference against SMAWM.

The WM baseline is trained in two stages: First, we train a variational autoencoder (VAE) with a 5-dimensional latent space, where the encoder and decoder are MLPs with the same architecture as the encoder and decoder used in SMAWM model. Second, we freeze the model parameters of the VAE and train a transition model in the form of a mixture density recurrent neural network (MDRNN) with mean-squared error on the latent representations, with a 10-dimensional temporal latent state and 3 components in the Gaussian mixture model. The spatial latent state learnt by the VAE and the temporal latent state learnt by the MDRNN are concatenated together to form the state representation to train the reinforcement learning agent. The number of parameters used in the WM baseline is 2434, explicitly designed to be significantly more than that of SMAWM at 1590.

6 Results



(a) Random Actions Setting (b) Spurious Correlations Setting (c) Expert Demonstrations Setting

Figure 3: The performance of the agent trained with SMAWM is significantly higher than WM across all settings of interest. Each point represents the mean reward from 1 random seed and the 95% confidence interval of the estimated mean is displayed.

SMAWM outperforms WM in all settings of interest, despite having significantly fewer trainable parameters (1590 vs 2434) and thus lower model expressiveness. Two reasons possibly explain the higher performance. First, with fewer trainable parameters, the SMAWM has a smaller solution space to search and thus is easier to train. Second, the graph-structured state representation learnt by the GNN-based transition model in SMAWM and its agent-factored nature is shown to be useful to model agent interactions in the multi-agent setting. We show in Section 7 that the superior state representation explains the improved performance well. Regardless, we show empirically that the state representation learnt by SMAWM is superior for downstream agent performance in all settings.

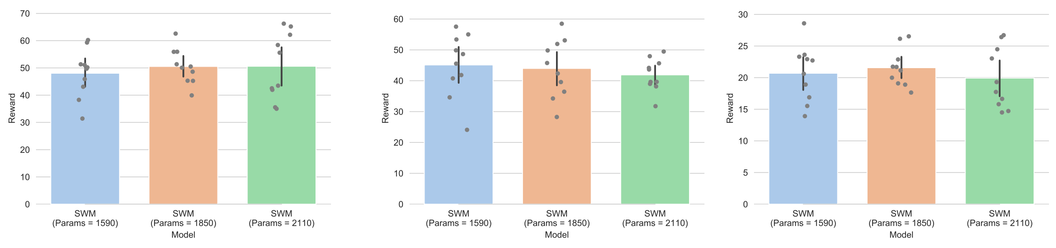
Setting	WM	SMAWM	Improvement
Random Actions	31.529	48.504	**+16.525**
Spurious Correlations	30.963	45.128	**+14.165**
Expert Demonstrations	18.717	20.699	+1.982

Table 1: Performance of agent trained with WM versus SMAWM over 10 random seeds.

Unexpectedly, SMAWM does not significantly outperform WM on the Spurious Correlations setting (+16.525) compared to the Random Actions setting (+14.165). We hypothesise that since the dataset contains 1000 episodes with 100 time steps each, a 95% correlation of actions between agents still results in 5000 samples that serve as non-spuriously correlated samples, which is possibly sufficiently representative for modeling transition dynamics over a combination of actions. Further experiments with fewer samples sizes should determine whether the SMAWM does indeed outperform WM in a spurious correlation setting.

7 Analysis

7.1 Parameter Count on Performance



(a) Random Actions Setting (b) Spurious Correlations Setting (c) Expert Demonstrations Setting

Figure 4: The performance of the agent trained with SMAWM does not significantly change as number of parameters increase.

As previously discussed in section 6, to determine whether the performance gain is simply due to a smaller solution search space or the effectiveness of an agent-factored state representation, we gradually increase the number of trainable parameters in SMAWM towards that of WM, and inspect any change in model performance. We increase the number of hidden layers to 3 to obtain a 1850-parameter SMAWM, and increase the latent dimension per agent to 10 to obtain a 2110-parameter SMAWM.

If a smaller solution space is the reason for higher performance, we should expect performance to decrease as number of parameters increase. However, as the number of parameters increase from 1590 to 1850 and 2110, there is no significant change in performance, showing that the performance improvements conferred by the SMAWM are indeed a result of the agent-factored state representation learnt.

Nonetheless, the lack of increase in performance raises an important question: Is the lack of performance improvement due to the simplicity of the problem or is it that the SMAWM is limited in performance improvements even after increasing model expressiveness? A similar lack of correlation between a higher number of parameters and higher performance has been observed in other works [6; 24], and we leave further diagnosis for future work.

7.2 Prediction Errors for Planning

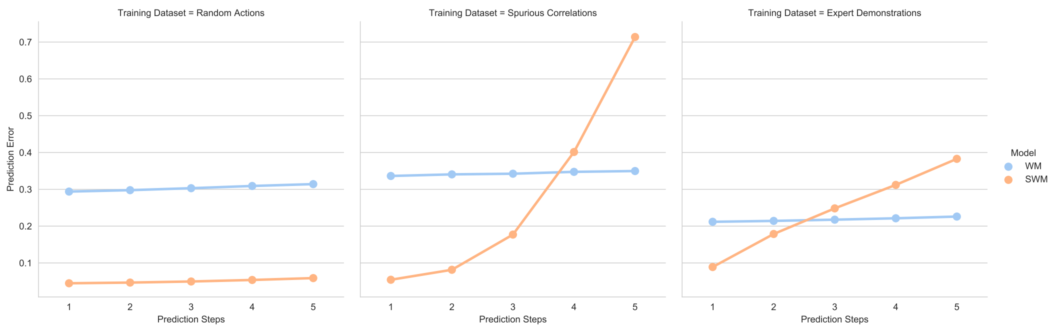


Figure 5: Prediction Errors over Time Steps

In model-based state-space planning, an agent utilises a model of the environment to predict forward into the future states given a set of actions [29]. Planning with a learnt model has proven essential for high performance across a range of problem domains and tasks [9; 10; 25]. To study the effectiveness of SMAWM for planning, we sample transition tuples from the Expert Demonstrations setting $\{(o_t, a_t, o_{t+1})\}_{t=0}^{T}$ where $T = 5$, and perform forward prediction with SMAWM and WM trained in the Random Actions, Spurious Correlations and Expert Demonstrations settings. Given $\{o_0, a_0, a_1, a_2, a_3, a_4\}$, the world models recursively apply 1-step forward prediction from the initial state o_0 to predict future observations $\{\hat{o}_1, \hat{o}_2, \hat{o}_3, \hat{o}_4, \hat{o}_5\}$. Prediction errors are computed as the mean squared error between predicted and actual observations.

Across all settings, the SMAWM has significantly lower prediction error for very short (1 to 2) time steps in forward prediction. However, as the number of time steps increase beyond 3, prediction errors of SMAWM quickly compound to exceed that of WM. It is particularly surprising to note that even though we expect SMAWM to outperform WM on the Spurious Correlations setting to a greater extent than on the Random Actions setting, prediction errors of SMAWM easily exceeds that of WM after 3 time steps in the Spurious Correlations setting while remaining way lower in the Random Actions Setting, suggesting that SMAWM as it currently is suffers more from spurious correlations than WM. We hypothesise one possible reason as the Markov assumption that the current state and action contain all the information necessary to predict the next state, which does not hold since the state does not capture velocity as simulated in the environment, and leave investigation for future work.

8 Limitations and Future Extensions

Opponent Modeling Even though the original idea includes an opponent model $a_{2:k} = f_\theta(s)$ where s represents the state and $a_{2:k}$ represents the actions taken by all other agents in the environment, due to insufficient time, we were unable to implement and study this component. The opponent model component is also essential to improve the novelty of this research project. In an intended future extension of this work, we will study the benefit of an opponent model in the full version of SMAWM.

The opponent model will take the form of a MLP $a_{2:k} = f(s_t)$. The opponent model predicts the actions $a_{2:k}$ that would be taken by all other $k-1$ agents. The transition model predicts the next state s_{t+1} given predicted agents' actions $\hat{a}_2, ..., \hat{a}_k$ and the agent's action a_1. The opponent model could take other forms, such as GNNs, designs from inverse RL, or from multi-agent RL like Self-Other Modeling (SOM) [22].

Stochasticity and Markov Assumption As a formulation inspired by C-SWMs [14], its limitations are similarly inherited. As it is, SMAWM does not take into account stochasticity in environment transitions. Future work should explore probabilistic extensions. In addition, since it currently assumes the Markov assumption, future work should study the incorporation of memory mechanisms such as recurrent neural networks, which is likely to improve planning accuracy. The transition model can take the form of a graph-structured variational recurrent neural network (Graph-VRNN), that models the transition dynamics and can be used in a partially observable setting [28].

Learning and Planning by Imagination SMAWM is currently utilised only as a feature extractor. Future work should study training agents purely within SMAWM without any real-world observations, as inspired by a closely-related line of work of world models [8; 9; 10]. SMAWM would need to be iteratively and actively updated as the agent explores in the environment and visits new states. SMAWM can also be used for online planning in latent space with Model Predictive Control [10].

9 Conclusions

We present Structured Multi-Agent World Models (SMAWM), a graph-structured formulation of agent-factored state representations using graph neural networks to model environment dynamics and agent interactions for MBRL in the multi-agent mixed cooperative-competitive setting. We show that reinforcement learning with the agent-factored state representation learnt by SMAWM in a multi-agent setting outperforms that with a purely connectionist world model despite using much fewer parameters in all settings of interest. We further show that SMAWM learns an effective representation that is capable of much higher accuracy in forward prediction for planning. Further research on incorporating opponent modeling, memory mechanisms and learning and planning by imagination can likely scale SMAWM to environments of higher complexity.

10 Acknowledgements

We would like to thank Muhammad Rizki Aulia Rahman Maulana for helpful discussions and Professor Leong Tze-Yun for the opportunity to do this research project in the scope of the CS4246 AI Planning and Decision Making class.

Contributions are as follows: New Jun Jie - Project formulation, experiments formulation, dataset generation, model implementations, experiment runs, results visualisation and analysis, presentation, report write-up, future work extensions; Wu Yujin - Dataset generation, error diagnosis, presentation, report write-up.

References

[1] Plaat Aske, Kosters Walter, and Mike Preuss. Model-based deep reinforcement learning for high-dimensional problems, a survey. *arXiv preprint arXiv:2008.05598*, 2020.

[2] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[3] Peter W Battaglia, Jessica B Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, et al. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*, 2018.

[4] Daan Bloembergen, Karl Tuyls, Daniel Hennes, and Michael Kaisers. Evolutionary dynamics of multi-agent learning: A survey. *Journal of Artificial Intelligence Research*, 53:659–697, 2015.

[5] Michael B Chang, Tomer Ullman, Antonio Torralba, and Joshua B Tenenbaum. A compositional object-based approach to learning physical dynamics. *arXiv preprint arXiv:1612.00341*, 2016.

[6] Gabriele Corso, Luca Cavalleri, Dominique Beaini, Pietro Liò, and Petar Veličković. Principal neighbourhood aggregation for graph nets. *arXiv preprint arXiv:2004.05718*, 2020.

[7] Sven Gronauer and Klaus Dieopold. Multi-agent deep reinforcement learning: a survey. *Artificial Intelligence Review*, pp. 1–49, 04 2021. doi: 10.1007/s10462-021-09996-w.

[8] David Ha and Jürgen Schmidhuber. World models. *arXiv preprint arXiv:1803.10122*, 2018.

[9] Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. *arXiv preprint arXiv:1912.01603*, 2019.

[10] Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. In *International Conference on Machine Learning*, pp. 2555–2565. PMLR, 2019.

[11] He He, Boyd-Graber Jordan, Kwok Kevin, and Hal Daume III. Opponent modeling in deep reinforcement learning. *arXiv preprint arXiv:1609.05559v1*, 2016.

[12] Yedid Hoshen. Vain: Attentional multi-agent predictive modeling. *arXiv preprint arXiv:1706.06122*, 2017.

[13] Thomas Kipf and Welling Max. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308*, 2016.

[14] Thomas Kipf, Elise van der Pol, and Max Welling. Contrastive learning of structured world models. *arXiv preprint arXiv:1911.12247*, 2019.

[15] Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. *arXiv preprint arXiv:1706.02275*, 2017.

[16] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pp. 1928–1937. PMLR, 2016.

[17] Thomas M Moerland, Joost Broekens, and Catholijn M Jonker. Model-based reinforcement learning: A survey. *arXiv preprint arXiv:2006.16712*, 2020.

[18] Constantin-Valentin Pal and Florin Leon. Brief survey of model-based reinforcement learning techniques. In *2020 24th International Conference on System Theory, Control and Computing (ICSTCC)*, pp. 92–97. IEEE, 2020.

[19] Aske Plaat, Walter Kosters, and Mike Preuss. Deep model-based reinforcement learning for high-dimensional problems, a survey. *arXiv preprint arXiv:2008.05598*, 2020.

[20] Aske Plaat, Walter Kosters, and Mike Preuss. High-accuracy model-based reinforcement learning, a survey. *arXiv preprint arXiv:2107.08241*, 2021.

[21] Athanasios S Polydoros and Lazaros Nalpantidis. Survey of model-based reinforcement learning: Applications on robotics. *Journal of Intelligent & Robotic Systems*, 86(2):153–173, 2017.

[22] Roberta Raileanu, Emily Denton, Arthur Szlam, and Rob Fergus. Modeling others using oneself in multi-agent reinforcement learning. In *International conference on machine learning*, pp. 4257–4266. PMLR, 2018.

[23] Alvaro Sanchez-Gonzalez, Nicolas Heess, Jost Tobias Springenberg, Josh Merel, Martin Riedmiller, Raia Hadsell, and Peter Battaglia. Graph networks as learnable physics engines for inference and control. In *International Conference on Machine Learning*, pp. 4470–4479. PMLR, 2018.

[24] Benjamin Sanchez-Lengeling, Emily Reif, Adam Pearce, and Alexander B. Wiltschko. A gentle introduction to graph neural networks. *Distill*, 2021. doi: 10.23915/distill.00033. https://distill.pub/2021/gnn-intro.

[25] Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, et al. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588(7839):604–609, 2020.

[26] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[27] Yoav Shoham, Rob Powers, and Trond Grenager. If multi-agent learning is the answer, what is the question? *Artificial intelligence*, 171(7):365–377, 2007.

[28] Chen Sun, Per Karlsson, Jiajun Wu, Joshua B Tenenbaum, and Kevin Murphy. Stochastic prediction of multi-agent interactions from partial observations. *arXiv preprint arXiv:1902.09641*, 2019.

[29] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[30] Fengji Yi, Wenlong Fu, and Huan Liang. Model-based reinforcement learning: A survey. In *Proceedings of the International Conference on Electronic Business (ICEB), Guilin, China*, pp. 2–6, 2018.

[31] Matej Zečević, Devendra Singh Dhami, Petar Veličković, and Kristian Kersting. Relating graph neural networks to structural causal models. *arXiv preprint arXiv:2109.04173*, 2021.