Transformer World Model for Sample Efficient Multi-Agent Reinforcement Learning

Azad Deihim[a], Eduardo Alonso[a], Dimitra Apostolopoulou[b]

[a] *Artificial Intelligence Research Centre (CitAI), City St George's, University of London*
[b] *Oxford Institute for Energy Studies (OIES), University of Oxford*

Abstract

We present the Multi-Agent Transformer World Model (MATWM), a novel transformer-based world model designed for multi-agent reinforcement learning in both vector- and image-based environments. MATWM combines a decentralized imagination framework with a semi-centralized critic and a teammate prediction module, enabling agents to model and anticipate the behavior of others under partial observability. To address non-stationarity, we incorporate a prioritized replay mechanism that trains the world model on recent experiences, allowing it to adapt to agents' evolving policies. We evaluated MATWM on a broad suite of benchmarks, including the StarCraft Multi-Agent Challenge, PettingZoo, and MeltingPot. MATWM achieves state-of-the-art performance, outperforming both model-free and prior world model approaches, while demonstrating strong sample efficiency, achieving near-optimal performance in as few as 50K environment interactions. Ablation studies confirm the impact of each component, with substantial gains in coordination-heavy tasks.

Keywords: Multi-Agent Reinforcement Learning, World Models, Transformers, Model-Based Reinforcement Learning, Representation Learning

1. Introduction

Reinforcement learning (RL) is a framework in which agents learn to make decisions by interacting with an environment to maximize cumulative rewards, typically through trial and error. Deep RL (DRL) extends this by using deep neural networks to approximate policies and value functions, enabling agents to operate in high-dimensional or complex environments. Within DRL, model-free algorithms refer to methods that learn directly from experience without constructing an explicit model of the environment's dynamics. These include popular approaches like Q-learning and policy gradients. Model-free DRL methods have led to breakthroughs in many DRL settings [1, 2, 3, 4, 5, 6, 7], but they often suffer from sample inefficiency, requiring millions of interactions to learn effective behaviors, making them impractical in environments where data collection is expensive or risky [8]. This efficiency gap has limited the use of RL in real-world tasks where large-scale interaction is not feasible. World models, however, offer a promising solution to this issue. In RL, a world model is a learned representation of the environment that allows agents to simulate

and learn from predicted future outcomes without interacting with the real environment [9]. By enabling agents to "imagine" future trajectories given some contextual observations and actions, they can compute and learn from up to 16 imagined future steps per real sample. A world model typically uses neural networks to replicate environment dynamics like transitions and rewards. This capability dramatically improves sample efficiency, which can make DRL more attractive for domains where real-world sampling of millions of experiences is impractical, such as robotics, healthcare, or autonomous driving systems.

While world models have demonstrated substantial improvements in sample efficiency for single-agent RL, their application to multi-agent environments remains largely unexplored due to the added complexity of modeling interactions among agents. Only a few existing frameworks, such as MBVD [10], MAMBA [11], and MARIE [12], have attempted to bring world modeling into multi-agent RL (MARL), and although these models achieve promising results, they still fall short of matching the complexity needed to tackle environments on par with those addressed by their single-agent counterparts, such as vast 3D environments like Minecraft [13].

In multi-agent settings, accurately predicting long-horizon trajectories becomes increasingly difficult, as the interactions of multiple agents introduce additional sources of error that can compound and propagate throughout the entire imagination rollout. As a result, horizon lengths that are feasible in single-agent settings often become infeasible in multi-agent environments. A further challenge stems from two strategies for modeling multi-agent dynamics: centralized and decentralized. A centralized world model approach may better capture agent interdependencies by using all agent information to compute a joint trajectory, but it scales poorly as the number of agents grows, leading to increased spatial complexity. A decentralized world model approach that only computes local trajectories based on local information, while more scalable, suffers from non-stationarity issues due to unpredictable and constantly evolving interactions with other agents, resulting in an inaccurate representation of imagined state transitions. However, a major advantage of using world models in multi-agent systems is the ability to apply new dynamics to old experiences. Generally, learning from older experiences has the potential to become harmful because they reflect outdated behaviors from other agents, but a world model with the capacity to track and update with each agent's evolving behavior can reinterpret old experiences given new dynamics, making old transitions more relevant.

To advance the capabilities of multi-agent world models, we introduce a novel Multi-Agent Transformer World Model (MATWM)[1] framework, built upon and inspired by STORM [14], a leading transformer world model architecture for single-agent systems. We selected STORM as the foundation for MATWM because it consolidates a wide range of best practices and architectural advances found across leading single-agent world models [15, 16, 17], such as transformer-based sequence modeling [18], Kullback–Leibler (KL) balance and free bits [19], percentile return normalization, two-hot symlog reward targets, discrete latent representations, and many more — several of which have not yet been incorporated into

[1] github.com/azaddeihim/matwm

any existing multi-agent world model architecture. This makes it an ideal starting point for developing a more complex and capable multi-agent world model, allowing us to focus our contributions on the novel extensions required for multi-agent imagination and coordination, such as teammate prediction, prioritized replay sampling, and action space scaling to distinguish between agents. We utilize a decentralized world model approach, and our agent training is semi-centralized, in that it does not explicitly use information from other agents during training, but instead uses imagined behavior of other agents during training. This follows the Centralized Training and Decentralized Execution (CTDE) design, but bypasses the need to have access to all agents' information during agent training.

The contributions of MATWM are as follows:

1. Solve complex multi-agent benchmarks, such as the StarCraft Multi-Agent Challenge (SMAC), PettingZoo, and MeltingPot, with as few as 50K real environment interactions, which, to our knowledge, is the lowest sample allowance of any existing MARL algorithm.
2. We incorporate several methods from top-performing single-agent world model architectures that have not yet been adopted in multi-agent settings, including but not limited to free bits, percentile return normalization, and two-hot symlog reward targets [16, 14].
3. We design a lightweight and effective teammate predictor module for our world model to model agents' behavior to improve cooperation between agents, where communication is otherwise not possible.
4. We employ a prioritized replay buffer to ensure the world model is trained on more recent and informative experiences, enabling it to stay aligned with the agents' evolving policies. Additionally, we use an action scaling mechanism to encode agent-specific information, helping the world model differentiate between individual agents.
5. We present, to our knowledge, the first multi-agent world model capable of learning from image-based observations. Unlike prior work, which has been limited to low-dimensional vector inputs, MATWM supports visual environments and demonstrates strong performance in this setting.

The remainder of this paper is structured as follows. In Section 2, we review prior work on world models and their extension to MARL. Section 3 details the architecture and training process of MATWM, including our key design choices and novel components. Section 4 presents experimental results across SMAC, PettingZoo, and MeltingPot environments, demonstrating the effectiveness and efficiency of our approach and analyzing the contribution of individual components via ablation studies. Finally, Section 5 summarizes our contributions and outlines directions for future work.

2. Related Work

World models have shown significant promise in improving sample efficiency through imagination-based training [9, 20, 21]. Recent efforts proposed to improve sample efficiency

in model-free algorithms have been successful through the use of auxiliary objectives [22, 23] or data augmentation [24, 25]. However, even with these improvements, model-free algorithms still require significantly more training samples to achieve results similar to those of world model-based algorithms.

The original world model [26] pioneered the idea in the single-agent setting, learning models for imagining future trajectories using autoencoders and recurrent neural networks (RNN) [27]. Following this, several improvements were made in imagined state representations, agent training, and structure and contents of the world model through frameworks such as SimPLE [28] and the Dreamer series [29, 15, 16]. Following the debut of Dreamer, several new world model architectures, IRIS [30], TWM [17], TransDreamer [31], and STORM [14], were able to further enhance the Dreamer method with the use of transformers as the core sequence model rather than GRU[32] or other RNN-based architectures, as transformers[18] typically outperform RNNs in a wide variety of sequence modeling tasks due to their ability to model long-range dependencies and enable parallel computation. However, all of these world model architectures only considered single-agent systems and would likely require significant rework to enable success in multi-agent settings.

The existing literature on multi-agent world models is very limited. MAZero [33], a multi-agent extension of MuZero [34], is a model-based approach to MARL that closely resembles world models; it uses Monte Carlo Tree Search to plan future trajectories instead of a traditional neural network-based world model, which was computationally expensive and scaled poorly with an increased number of agents or larger action spaces. MBVD and MAMBA were among the first model-based approaches to employ neural network-based world models in MARL. MBVD leverages shared latent rollouts and value decomposition to facilitate coordination, utilizing a GRU-based sequence model [32] to generate imagined trajectories. While it introduced an important architectural step forward as one of the earliest world models tailored for multi-agent systems, it still suffers from low sample efficiency, often requiring over one million environment steps to reach performance levels that more recent multi-agent world model methods can achieve in under 100,000 steps. MAMBA, a multi-agent extension of DreamerV2, introduces a centralized world model approach that uses joint-agent dynamics to compute a joint-agent state using a GRU-based sequence model, enabling imagination-based learning in partially observable environments. It has demonstrated strong performance across a range of challenging MARL tasks, including those with high agent counts and partial observability, outperforming traditional model-free approaches and earlier model-based methods. While effective for modelling interactions between agents, a downside of the centralized world model can become extremely computationally expensive as the number of agents grows.

At the time of writing, MARIE [12] introduces the first and only Transformer-based world model for multi-agent systems, which achieved a substantial improvement over MBVD and a marginal improvement MAMBA in a wide range of SMAC maps. It learns decentralized local dynamics alongside centralized feature aggregation via a Perceiver [35]. The aggregation module is used to compute contextual information about the entire team from a given agent's perspective. These aggregated features are inserted into each agent's local

token sequence and passed to the shared dynamics model, allowing for coordination and mitigating non-stationarity during imagination. This method proved successful in enabling multi-agent imagination while being far more memory-efficient than the centralized world model approach found in MAMBA. Our approach shares the goal of enabling sample efficient multi-agent learning through imagination but takes a complementary route: instead of modeling all agent interactions through centralized aggregation, we explicitly model other agents' behavior with a learned teammate predictor. The computational cost of the teammate predictor remains nearly constant as the number of agents increases, whereas the aggregation module's computational cost does increase with the number of agents.

MARIE, along with MAMBA, use PPO-style training [36] for their world model and agents; this can be advantageous in multi-agent systems as it ensures that experiences being used to train the world model and agents are recent and do not reflect outdated agent behaviours, but PPO-style training is generally not as sample efficient due to infrequent updates and can suffer from catastrophic forgetting [37]. Instead, we adopt a DreamerV3-style [16] training framework augmented with a prioritized replay mechanism that emphasizes recent experiences. This setup enables more frequent updates, training both the world model and the agent after each environment step. We hypothesize that this approach enhances sample efficiency due to the higher volume of updates, while mitigating overfitting in agent training by continuously updating imagined trajectories in sync with evolving policies. Additionally, we extend our framework to support both image-based and vector-based observations, whereas prior multi-agent world models have been limited to vector-based inputs only.

2.1. Latent Representations

One of the primary distinctions between world model implementations lies in how state information is represented and provided to the agent. Training directly on raw observations can introduce significant noise and dramatically increase computational requirements [38, 15, 17], especially so for image data. To address this, most world models now rely on learned latent representations to encode essential features while discarding irrelevant noise and decreasing the dimensionality of the data. Early world model approaches typically employed continuous latent spaces, most commonly learned via standard variational autoencoders (VAEs) [39]. As opposed to the Vector Quantized Variational Autoencoder (VQ-VAE), which is a self-supervised method that learns discrete latent representations, achieving comparable performance while producing significantly more compact encodings [40]; for example, when applied to environments like DeepMind Lab [41], they can represent observations using as few as 27 bits. Furthermore, empirical studies have shown that discrete latent spaces often outperform continuous ones on a wide range of RL tasks, even in model-free algorithms [42].

Nonetheless, DreamerV1 [29] and World Models [26] adopted continuous latent spaces for their frameworks. SimPLE [28] was among the first to instead utilize a VQ-VAE. However, this approach was soon surpassed by DreamerV2 [15], which adopted a Categorical-VAE [43] that achieved substantially better performance. Today, several of the most competitive single-agent world models employ Categorical-VAEs as their default. In contrast,

Categorical-VAEs are less prevalent in multi-agent world models. MBVD utilizes continuous latent spaces, whereas MARIE employs discrete representations through a VQ-VAE. MAMBA, however, does use a Categorical-VAE. We hypothesize that our use of a Categorical-VAE will provide a slight representational advantage over the existing multi-agent world models that have not yet adopted it.

We present a comparative overview of MARIE, MAMBA, MBVD, and MATWM in Table 1, highlighting key distinctions in architectural and training design.

Table 1: Comparison of MATWM with leading multi-agent world model architectures

Aspect	**MARIE**	**MAMBA**	**MBVD**	**MATWM**
Multi-Agent Design	Yes	Yes	Yes	Yes
Backbone Architecture	Transformer	GRU	GRU	Transformer
Latent Representation	VQ-VAE	Categorical VAE	VAE	Categorical VAE
Critic Type	Centralized	Centralized	Centralized	Semi-centralized
World Model Type	Hybrid	Centralized	Centralized	Decentralized
Visual Observation Support	No	No	No	Yes
Agent Training Strategy	PPO-style	PPO-style	Deep Q-learning	DreamerV3-style

3. Method

All agents in the environment utilize a shared world model, which is trained using an equal distribution of experiences from each agent. In this paper, the focal agent refers to the agent currently being trained, evaluated, or associated with a given experience, or for whom the world model is generating predictions and trajectories. Our approach builds upon STORM [14], a single-agent world model architecture that integrates several state-of-the-art design principles, including transformer-based sequence modeling, symlog reward scaling, KL balance and free bits, and categorical latent representations. MATWM retains STORM's core components but adds key extensions for multi-agent settings. Specifically, MATWM adds a teammate predictor to model other agents' behavior, a prioritized replay sampling strategy to maintain up-to-date world model training, and action scaling mechanisms that allow agents to be distinguished within shared environments. These additions enable MATWM to support multi-agent coordination without requiring centralized training or communication. The framework utilized in this paper is consistent with that of other world model-based algorithms; agents learn a policy via imaginations generated by the world model. This is done via the following three steps that repeat in order:

1. Populate a replay buffer with real experiences using the current policies of the agents.
2. Train the world model using samples from the replay buffer.
3. Sample real experiences from the replay buffer, compute imagined trajectories using the real experiences as a starting point, and update the agents' policies using the imagined trajectories.

These steps are iterated until the maximum number of allowed steps through the real environment is achieved. Each entry in the replay buffer contains an observation o_t, an

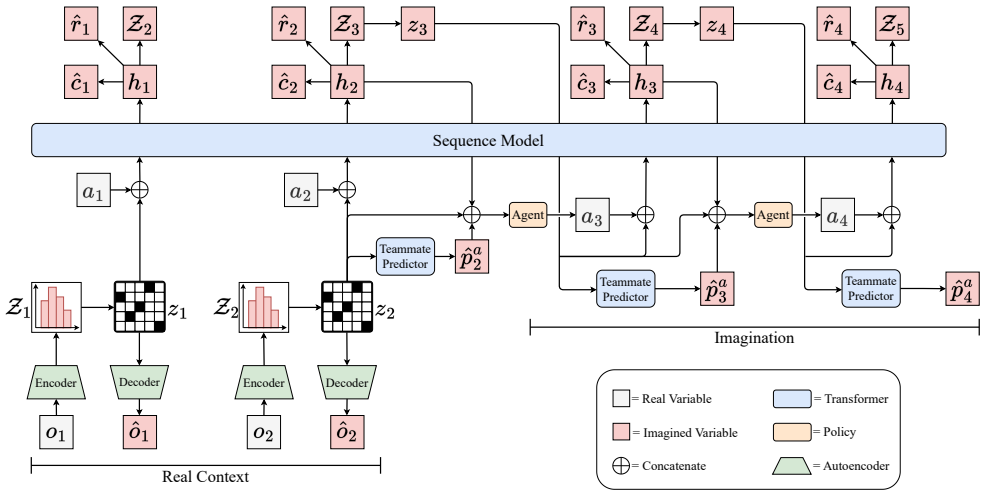

Figure 1: Overview of the MATWM architecture. The figure illustrates how real context (past observations and actions) are encoded and used as a starting point for imagination rollouts. Each agent encodes its local observation into a discrete latent state via an autoencoder. These latent states, combined with actions, are processed through an action mixer and a transformer-based sequence model to produce hidden states. The world model then predicts the next latent state, reward, continuation, and optionally, the action mask. The agent then uses the new latent state, predicted teammate action logits and the hidden state to compute a new action. The new action and latent state are then used to repeat this process.

action a_t, a reward r_t, a continuation flag c_t (a Boolean variable denoting whether an episode has terminated), and an action mask m_t, if applicable. All agents will have their own replay buffer. Agents' actions will be randomly sampled until the replay buffer reaches 1,000 samples, after which agents and the world model will begin training, and agents will use their learned policy to decide actions. After each real environment step, the world model and the agents will undergo one epoch of training. For world model training, we sample a batch of 64-length sequences from the replay buffer, prioritizing recent samples using an exponentially decaying weighting system. We also ensure that there is no overlap between sequences to promote diversity in the training batch. When sampling from the replay buffer for agent training, we select experiences entirely at random without any weighting. We utilize a novel trick for agent-world model interactions by scaling each agent's action space to be mutually orthogonal. For instance, if one agent's discrete action space is {0, 1, 2}, the next agent's actions are offset to {3, 4, 5}, and so on for additional agents. This enables the world model to distinguish agents without requiring explicit IDs or embeddings.

3.1. World Model

The world model is a collection of predictive models that can replicate the dynamics of the real environment, allowing the agents to interact with and learn from the world model, rather than the real environment directly. Our world model architecture comprises several modules that enable the accurate replication of the real environment's dynamics. Figure

1 shows a diagram of the MATWM architecture and how a focal agent interacts with it, further explained in Algorithm 1 found in Appendix A. We build a world model that is shared by all agents, but only computes the focal agents trajectory using only the focal agent's information.

We use an autoencoder to transform each observation o_t into a stochastic latent categorical distribution $\mathcal{Z}_t$. Following prior work [15, 14, 16], we define $\mathcal{Z}_t$ as consisting of 32 categorical variables, each with 32 discrete classes. The encoder q_ϕ and decoder p_ϕ are implemented as convolutional neural networks (CNNs) [44] for image observations and multi-layer perceptrons (MLPs) for vector observations. The latent state z_t is sampled from $\mathcal{Z}_t$ to represent the original observation o_t. Since sampling from a categorical distribution is non-differentiable, we use the straight-through gradient estimator [45] to preserve gradients for backpropagation. We define the encoder and decoder as follows:

$$\begin{aligned} &\text{Encoder:} \quad z_t \sim q_\phi\left(z_t \mid o_t\right) = \mathcal{Z}_t \\ &\text{Decoder:} \quad \hat{o}_t = p_\phi\left(z_t\right) \end{aligned} \tag{1}$$

We utilize an action mixer, a single linear transformation denoted by the function $m_\phi(\cdot)$, that merges the latent state z_t with the agent's action a_t, returning an embedded representation e_t. This is passed into a vanilla transformer [18], whose function is denoted by $f_\phi(\cdot)$, acting as the sequence model, which outputs the hidden states $h_{0:T}$ for all $e_{0:T}$. Then, the dynamics predictor, an MLP-based module, denoted by the function $g_\phi(\cdot)$, predicts the next latent state $\hat{z}_{t+1}$ from the h_t. Since no real observations are available during imagination, the dynamics predictor is critical for rolling out future states, as we would not be able to use the encoder to compute them. The reward and continuation predictors estimate rewards and continuation flags during imagination. We define these world model components as:

$$\begin{aligned} &\text{Action mixer:} && e_t = m_\phi^E\left(z_t, a_t\right) \\ &\text{Sequence model:} && h_{0:T} = f_\phi\left(e_{0:T}\right) \\ &\text{Dynamics predictor:} && \hat{\mathcal{Z}}_{t+1} = g_\phi^D\left(\hat{z}_{t+1} \mid h_t\right) \\ &\text{Teammate predictor:} && \hat{p}_{t,0}^{(a)}, \cdots, \hat{p}_{t,N}^{(a)} = f_\phi(z_{0:T}) \\ &\text{Reward predictor:} && \hat{r}_t = g_\phi^R\left(h_t\right) \\ &\text{Continuation predictor:} && \hat{c}_t = g_\phi^C\left(h_t\right) \\ &\text{Action Mask predictor:} && \hat{m}_t = g_\phi^M\left(z_t\right) \end{aligned} \tag{2}$$

To account for the behaviors of other agents and enable cooperative behavior during both imagination and interaction with the real environment, the teammate predictor utilizes a transformer to infer the actions of all N non-focal agents based on the focal agent's latent state history. Lastly, the action mask predictor is critical in partially observable or constrained environments like SMAC, where agents are only allowed to execute legal actions. Since imagined rollouts do not have access to real action masks, we train a predictor to estimate them. This allows us to avoid degenerate scenarios where an agent conducts an

illegal action, for which the sequence model may produce an invalid state, thus tainting the entire imagination rollout.

For each agent in the environment, we randomly sample transitions from their replay buffer, then use them to update the world model via the total world model loss function:

$$\mathcal{L}(\phi) = \frac{1}{BT} \sum_{n=1}^{B} \sum_{t=1}^{T} \Big[\mathcal{L}_t^{\text{rec}}(\phi) + \mathcal{L}_t^{\text{rew}}(\phi) + \mathcal{L}_t^{\text{con}}(\phi) + \mathcal{L}_t^{\text{team}}(\phi) + \beta_1 (\mathcal{L}_t^{\text{mask}}(\phi) + \mathcal{L}_t^{\text{dyn}}(\phi)) + \beta_2 \mathcal{L}_t^{\text{rep}}(\phi) \Big], \tag{3}$$

where B is the batch size, T is the batch length, and $\mathcal{L}_t^{\text{rec}}$, $\mathcal{L}_t^{\text{rew}}$, $\mathcal{L}_t^{\text{con}}$, $\mathcal{L}_t^{\text{team}}$, $\mathcal{L}_t^{\text{mask}}$, $\mathcal{L}_t^{\text{dyn}}$, and $\mathcal{L}_t^{\text{rep}}$ are sub-loss functions for different subsets of the world models components.

The reconstruction loss, $\mathcal{L}_t^{\text{rec}}$, is represented by the mean-squared error between the real observation and the predicted reconstruction of the observation, facilitating the training of the encoder and the decoder:

$$\mathcal{L}_t^{\text{rec}}(\phi) = (\hat{o}_t - o_t)^2. \tag{4}$$

Next, $\mathcal{L}_t^{\text{rew}}$ and $\mathcal{L}_t^{\text{con}}$ are the components of the world model loss that facilitate the training of the reward and continue predictors, respectively. In $\mathcal{L}_t^{\text{rew}}$, we utilize a symlog two-hot loss, as described in [16]. $\mathcal{L}_t^{\text{con}}$ is the binary cross-entropy loss between the ground truth continuation flag c_t and predicted continuation flag $\hat{c}_t$. These sub-loss functions are computed as follows:

$$\mathcal{L}_t^{\text{rew}}(\phi) = \mathcal{L}^{\text{sym}}(\hat{r}_t, r_t), \tag{5}$$

$$\mathcal{L}_t^{\text{con}}(\phi) = c_t \log \hat{c}_t + (1 - c_t) \log (1 - \hat{c}_t). \tag{6}$$

The action mask loss $\mathcal{L}_t^{\text{mask}}$ is computed via the binary cross-entropy between the predicted action mask $\hat{m}_t$ and the ground-truth binary action mask m_t:

$$\mathcal{L}_t^{\text{mask}}(\phi) = m_t \log \hat{m}_t + (1 - m_t) \log (1 - \hat{m}_t). \tag{7}$$

The teammate predictor, given the latent state sequence of the focal agent, outputs a logits representing the probability of each teammate's possible actions. The ground truth teammate actions are used to compute a standard cross-entropy loss. For each teammate $i \in 0, \cdots, N$ and each possible action $a \in 0, \cdots, A$, where A is the number of discrete actions, we define:

$$\mathcal{L}_t^{\text{team}}(\phi) = -\sum_{i=1}^{N} \sum_{a=1}^{A} \delta(a_{t,i} = a) \log \hat{p}_{t,i}^{(a)}, \tag{8}$$

where $a_{t,i}$ is the ground truth action of teammate i at time t, and $\hat{p}_{t,i}^{(a)}$ is the predicted probability of teammate i selecting action a. During training, the latent state input to the teammate predictor is given as $\text{sg}(z_{0:T})$, where $\text{sg}(\cdot)$ denotes the stop-gradient operator. This is important as the actions can be noisy and random due to undertraining and/or entropy, and we do not want this random noise to backpropagate to the encoder.

The sub-loss functions $\mathcal{L}_t^{\text{dyn}}$ and $\mathcal{L}_t^{\text{rep}}$ are both represented as Kullback–Leibler (KL) divergences, and only differ in the placement of the stop-gradient operator, $\text{sg}(\cdot)$. Here, $\mathcal{L}_t^{\text{dyn}}$ aims to train the sequence model to predict the next latent state accurately. However, $\mathcal{L}_t^{\text{rep}}$ incentivizes the encoder to compute latent states that are easier to predict correctly. We follow [16, 14] in employing free bits [19] for these sub-loss functions to effectively disable them when they are sufficiently minimized, allowing the main objective loss function to focus on other sub-losses. This also mitigates trivial dynamics that are easy to predict but lack sufficient input information. These sub-losses are given by:

$$\mathcal{L}_t^{\text{dyn}}(\phi) = \max\left(1, \text{KL}\left[\text{sg}\left(q_\phi\left(z_{t+1} \mid o_{t+1}\right)\right) \| g_\phi^D\left(\hat{z}_{t+1} \mid h_t\right)\right]\right), \tag{9a}$$

$$\mathcal{L}_t^{\text{rep}}(\phi) = \max\left(1, \text{KL}\left[q_\phi\left(z_{t+1} \mid o_{t+1}\right) \| \text{sg}\left(g_\phi^D\left(\hat{z}_{t+1} \mid h_t\right)\right)\right]\right). \tag{9b}$$

3.2. Training Structure

To train agents using imagined experience, we start from a context window of real observations and actions. The world model uses this context to produce an initial latent state and hidden state. During each imagination step, the focal agent samples an action based on its current state and the predicted action distribution of its teammates, produced by the teammate predictor. When enabled, the action mask predictor is used to enforce action feasibility constraints. The action is then passed to the world model to predict the next latent state, hidden state, reward, and termination signal, which are stored in temporary rollout buffers. The resulting trajectories, stored in that temporary rollout buffer, allow the agent policies to learn entirely from imagination, without requiring new environment interactions.

Each agent is trained in isolation via a shared decentralized world model. Agents do not interact with one another directly; instead, they interact with imagined versions of other agents whose behavior is captured within the state information and by the teammate predictor. The rationale behind this approach is that the world model will treat non-focal agents as it would any other non-deterministic entity, similar to the non-agent adversaries in a single-agent Atari game, predicting their behavior and thus impact on the following state(s) to the best of its ability. The alternative, having agents coexist in and interact with the same imagined environment, poses several challenges. Firstly, the hidden state for each agent h_t may encompass information from all agents, potentially leading to situations where agents receive and become reliant on information they are not meant to have, especially in partially observable environments. Secondly, computing a joint trajectory would cause the memory requirements of the world model to scale poorly with the number of agents, making it impractical for environments involving many agents. Nonetheless, we hypothesize that our training structure could exacerbate the non-stationarity issue, as the world model might be displaying outdated behaviors of the agents during imagination. Again, we alleviate this issue by using a replay memory structure that prioritizes sampling recent memories when training the world model. We share the full training algorithm in Appendix B.

The state s_t, for each agent, is formed by concatenating the predicted action logits $\hat{p}_t^{(a)}$ of all non-focal agents and its own z_t and h_t. Agents are trained through an actor-critic

learning algorithm. Each agent will have its own critic, which only takes in local state s_t of the critic's respective agent along with that agent's a_t. A centralized critic would not be feasible here because during imagination, the agents' states are very likely to become desynchronized, so the actions conducted by an agent in its imagined state may not be transferable to another agent's imagined state, even in the same time step; additionally, they may have imaged states that are conflicting and cannot legally exist at the same time. This is why we use the teammate predictor to predict what the non-focal agents would do in the focal agent's current state, allowing us to have a semi-centralized critic that does not require having direct access to non-focal agent information. The Critic and Actor are denoted by:

$$\begin{aligned} &\text{Critic}: \quad V_\psi\left(s_t\right) \approx \mathbb{E}_{\pi_\theta, p_\phi}\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k}\right], \\ &\text{Actor}: \quad a_t \sim \pi_\theta\left(a_t \mid s_t\right). \end{aligned} \tag{10}$$

$$\begin{aligned} \mathcal{L}(\theta) &= \frac{1}{BL} \sum_{n=1}^{B} \sum_{t=1}^{L} \left[-\operatorname{sg}\left(\frac{G_t^\lambda - V_\psi\left(s_t\right)}{\max(1, S)} \right) \ln \pi_\theta\left(a_t \mid s_t\right) - \eta H\left(\pi_\theta\left(a_t \mid s_t\right)\right) \right], \\ \mathcal{L}(\psi) &= \frac{1}{BL} \sum_{n=1}^{B} \sum_{t=1}^{L} \left[\left(V_\psi\left(s_t\right) - \operatorname{sg}\left(G_t^\lambda\right) \right)^2 + \left(V_\psi\left(s_t\right) - \operatorname{sg}\left(V_{\psi^{\mathrm{EMA}}}\left(s_t\right)\right) \right)^2 \right]. \end{aligned} \tag{11}$$

Where $H(\cdot)$ represents the entropy of the policy distribution, and the constants η and L denote the entropy regularization coefficient and the imagination horizon, respectively. The λ-return G_t^λ is recursively defined [46] over the imagination horizon as follows:

$$\begin{aligned} G_t^\lambda &\doteq r_t + \gamma c_t \left[(1-\lambda) V_\psi\left(s_{t+1}\right) + \lambda G_{t+1}^\lambda \right], \\ G_L^\lambda &\doteq V_\psi\left(s_L\right). \end{aligned} \tag{12}$$

The normalization factor S used in the actor loss (Equation 11) is defined in Equation (13). It is computed as the difference between the 95th and 5th percentiles of the λ-return G_t^λ within the batch [16]:

$$S = \operatorname{percentile}\left(G_t^\lambda, 95\right) - \operatorname{percentile}\left(G_t^\lambda, 5\right). \tag{13}$$

To regularize the value function, we maintain an exponential moving average (EMA) of the critic parameters ψ. As defined in Equation (14), ψ_t represents the current critic parameters, σ is the decay rate, and $\psi_{t+1}^{\mathrm{EMA}}$ denotes the updated EMA. This technique helps stabilize training and mitigate overfitting:

$$\psi_{t+1}^{\mathrm{EMA}} = \sigma \psi_t^{\mathrm{EMA}} + (1-\sigma)\psi_t. \tag{14}$$

4. Experiments

We evaluate MATWM on three benchmark suites: the StarCraft Multi-Agent Challenge[47] for vector-based environments, and PettingZoo [48] and MeltingPot[49] for image-based, cooperative multi-agent settings. SMAC provides partially observable micro-management scenarios in StarCraft II, where agents must coordinate to defeat built-in AI opponents. It includes scenarios (maps) of varying difficulty, with *easy* maps requiring basic coordination and *hard* maps generally involving tighter coordination and complex tactics. PettingZoo offers diverse multi-agent environments; we focus on PettingZoo Butterfly as those are both image-based and cooperative. MeltingPot targets social interaction, with visual environments that require cooperation, competition, and negotiation among agents; for this suite we select a set of representative cooperative environments.

Following prior work [47, 12], we report the evaluation median and standard deviation of each experiment over four random seeds. Evaluation is based on the mean win rate or mean reward of all agents across 50 episodes. For SMAC, we compare against three world models (MARIE, MAMBA, MBVD) and three model-free baselines across 12 maps (9 *easy*, 3 *hard*). *Easy* maps are trained for 50K steps, and *hard* maps for 200K. While 100K is the standard for *easy* maps, prior work has shown that many models already reach 100% win rate under that budget [12, 11], so we use 50K to mitigate multi-way ties [12, 11]. For PettingZoo and MeltingPot, we compare against four model-free methods, using a 50K-step training budget. Hyperparameter settings for this study are provided in Appendix C.

We compare MATWM against several state-of-the-art baselines in MARL. First, we evaluate it against three world model architectures: MARIE, MAMBA, and MBVD, all of which represent the current frontier in world models for multi-agent systems. In addition to these model-based baselines, we also compare MATWM with a set of strong model-free algorithms, providing a broad and representative benchmark across different learning paradigms. QMIX [4] is a popular value-decomposition method that factors the joint action-value function into individual agent utilities under a monotonicity constraint. QPLEX [5] extends QMIX by incorporating a duplex dueling architecture that enables more expressive value functions. MAPPO [3] is a centralized variant of Proximal Policy Optimization [36] adapted for multi-agent settings, using a shared critic across agents. MAA2C [50] (Multi-Agent Advantage Actor-Critic) is an early actor-critic method for MARL that uses decentralized policies with a shared critic to stabilize training. We do not include MAA2C in SMAC or other vector-based benchmarks, as we already compare against several strong model-free and model-based baselines. MAA2C is included in our image-based evaluations only (PettingZoo and MeltingPot) primarily to provide an additional relevant model-free baseline, since existing world model approaches are omitted from these studies. These model-free baselines are selected as they are widely-used strong performers across a variety of MARL benchmarks.

4.1. SMAC Results

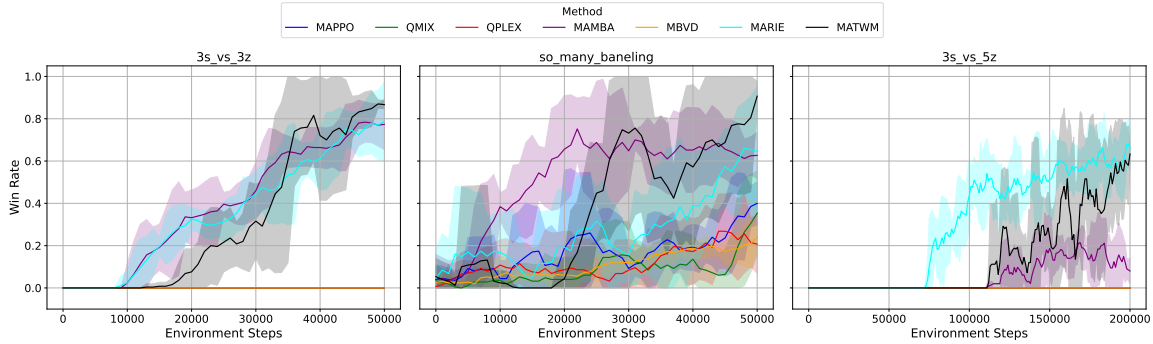

Figure 2: Curves of evaluation win rates for each method on three representative SMAC maps. Curves are smoothed for visual clarity of the graphs. The x-axis represents the number of steps taken in the environment, while the y-axis shows the corresponding evaluation win rate.

Table 2: Win rate as a % comparison across various SMAC maps: Median (Std).

Map	**Steps**	**MATWM**		**MARIE**		**MAMBA**		**MBVD**		**MAPPO**		**QMIX**		**QPLEX**	
		Med	**Std**	**Med**	**Std**	**Med**	**Std**	**Med**	**Std**	**Med**	**Std**	**Med**	**Std**	**Med**	**Std**
2m_vs_1z	50K	**98.0**	3.2	95.0	4.4	91.0	6.2	41.0	20.7	51.0	10.3	62.0	8.9	70.0	13.0
2s_vs_1sc	50K	**96.0**	5.7	90.0	9.1	80.0	7.3	0.0	1.2	18.0	7.6	8.0	4.4	13.0	6.8
2s3z	50K	**80.0**	9.0	71.0	8.6	68.0	12.1	28.0	17.5	13.0	3.0	17.0	4.3	36.0	5.8
3m	50K	**83.0**	10.4	78.0	14.1	68.0	7.7	60.0	9.2	54.0	6.3	61.0	10.2	58.0	4.9
3s_vs_3z	50K	**87.0**	19.4	85.0	21.8	77.0	23.7	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3s_vs_4z	50K	**12.0**	4.8	0.0	0.8	4.0	1.4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
8m	50K	67.0	24.9	**72.0**	7.1	68.0	6.4	52.0	18.9	38.0	4.9	60.0	18.2	59.0	9.4
MMM	50K	**7.0**	4.7	1.0	1.6	3.0	3.5	0.0	0.0	0.0	0.0	0.0	0.0	3.0	2.7
so_many_baneling	50K	**86.0**	22.9	73.0	12.4	66.0	14.2	27.0	12.3	31.0	7.6	40.0	6.1	52.0	8.8
3s_vs_5z	200K	64.0	26.5	**66.0**	28.0	6.0	10.1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2c_vs_64zg	200K	7.0	7.5	**14.0**	8.2	0.0	0.8	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
5m_vs_6m	200K	**46.0**	21.8	40.0	11.7	14.0	7.3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
Mean	—	**61.1**	—	57.1	—	45.4	—	17.3	—	17.1	—	20.7	—	24.3	—

We evaluate our framework on the StarCraft Multi-Agent Challenge, a widely adopted benchmark for multi-agent coordination under partial observability. Our model is compared against both model-free and model-based baselines across a diverse set of 12 scenarios, including nine *easy* maps and three *hard* maps. The results of these experiments are displayed in Table 2. We observe that, given a very small sample budget, MATWM outperforms all model-free baselines and even MBVD by a substantial margin, while also outperforming MAMBA and MARIE by lesser, but still decisive margins. On several *easy* maps, such as 2m_vs_1z, 2s_vs_1sc, 2s3z, and so_many_baneling, our model exhibits near-optimal behavior, achieving win rates of 85% or higher. In 3s_vs_3z, where tighter coordination is required, MATWM remains ahead, while model-free baselines fail to learn a winning strategy in the 50K sample budget. In 3s_vs_4z and MMM, the hardest of the *easy* maps, it is difficult for even the world model baselines to consistently learn a winning strategy within the sample

budget; MATWM, however, is able to do so more consistently than other world model baselines. Particularly in the *hard* maps, such as 2c_vs_64zg and 3s_vs_5z, model-free methods exhibit no success, unable to learn a winning strategy, while MATWM continues to be able to do so consistently. MARIE shows slight gains over our method on certain *hard* maps when given more training, but our model remains more sample efficient overall. Figure 2 shows evaluation win rates measured every 1,000 environment steps across three representative SMAC maps. Compared to other world model baselines, MATWM often requires more time to learn a winning strategy. However, once it does, it rapidly refines and exploits that strategy, achieving significantly higher win rates in a shorter span. These results establish our framework as a leading model in low-data, multi-agent settings, especially where coordination and partial observability are critical.

4.2. PettingZoo & MeltingPot

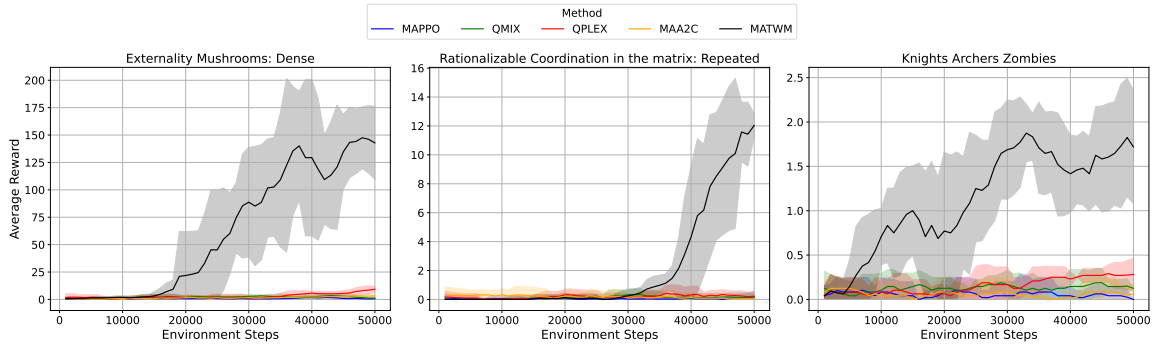

Figure 3: Curves of average reward obtained across all agents for each method on three representative PettingZoo and MeltingPot environments. Curves are smoothed for visual clarity of the graphs. The x-axis represents the number of steps taken in the environment, while the y-axis shows the corresponding average reward across all agents.

Table 3: Performance comparison across various PettingZoo Butterfly environments: Median (Std).

Environment	**Steps**	**MATWM**		**MAA2C**		**MAPPO**		**QMIX**		**QPLEX**	
		Med	**Std**	**Med**	**Std**	**Med**	**Std**	**Med**	**Std**	**Med**	**Std**
Cooperative Pong	50K	**54.5**	25.1	9.5	3.1	5.5	1.8	10.2	2.4	11.7	2.9
Knights Archers Zombies	50K	**1.75**	1.3	0.0	0.1	0.0	0.2	0.12	0.3	0.25	0.3
Pistonball	50K	**92.6**	2.3	2.5	0.0	-1.3	0.0	4.1	0.0	8.4	0.0

Table 4: Performance comparison across various MeltingPot Environments: Median (Std).

Environment	Steps	MATWM		MAA2C		MAPPO		QMIX		QPLEX	
		Med	Std	Med	Std	Med	Std	Med	Std	Med	Std
Chicken in the matrix: Arena	50K	**21.5**	2.4	1.3	0.8	2.1	1.1	3.1	1.4	2.7	0.9
Coop Mining	50K	**19.0**	4.1	9.5	6.1	1.4	0.7	1.0	0.6	1.7	0.5
Externality Mushrooms: Dense	50K	**146.8**	18.5	1.9	3.1	1.0	3.2	2.2	6.7	9.2	9.1
Gift Refinements	50K	**75.0**	25.1	9.5	6.1	6.5	10.8	14.2	16.4	23.7	21.9
Pure Coordination in the matrix: Repeated	50K	**6.8**	0.7	0.0	0.4	0.0	0.0	0.0	0.0	0.0	0.1
Rationalizable Coordination in the matrix: Repeated	50K	**12.2**	0.5	0.3	0.9	0.0	0.5	0.2	0.3	0.4	0.8
Stag Hunt in the matrix: Arena	50K	**7.2**	2.5	1.0	0.7	0.7	0.4	0.0	0.7	1.2	1.0

We evaluate the performance of MATWM on image-based multi-agent environments using the PettingZoo and MeltingPot benchmark suites. World model baselines are not included for PettingZoo and MeltingPot due to the absence of published adaptations or results for use in image-based environments. To the best of our knowledge, MATWM is the first world model framework evaluated in this domain. The results in Tables 3 and 4 highlight the strong generalization and sample efficiency of our model in visually complex settings. In the PettingZoo scenarios, MATWM consistently outperforms all four model-free baselines across all evaluated tasks. For instance, despite Pistonball's staggering 20 agents, MATWM is still able to achieve near-optimal performance with a very small sample budget, outperforming model-free baselines by a significant margin. We observe a similar trend in Cooperative Pong and Knights Archers Zombies, a substantial margin over baselines, whereas the model-free baselines perform roughly similar to a random policy. This showcases our model's capacity to learn effective cooperative behaviors from visual observations under tight sample constraints. MeltingPot results further reinforce these trends. Our model consistently achieves high rewards across all scenarios, regardless of the number of agents, which range between two and eight, compared to scores from all baselines. Even in more subtle or sparse-reward environments like Pure Coordination in the matrix: Repeated, MATWM is still able to quickly learn strategies that achieve high rewards. Similarly to the PettingZoo experiments, model-free baselines behave at or only slightly better than a random policy on nearly all evaluated environments, highlighting their inability to learn meaningful behavior under limited training budgets. Previous studies conducted on PettingZoo Butterfly environments suggest that it would take nearly one million or more environment steps for any of the model-free baselines to achieve comparable results to what MATWM did in just 50k [51, 52]. Figure 3 shows performance curves for MATWM and baseline methods evaluated at after every 1000 environment steps across three representative environments: one from PettingZoo and two from MeltingPot. These results demonstrate the benefits of incorporating imagined trajectories for sample efficiency as baselines often fail to make progress. One recurring pattern we observed during training, also visible in the figure, is that MATWM's performance occasionally dips before stabilizing again. Although this may be environment dependent, we hypothesize that such fluctuations may actually stem from the world model temporarily falling behind the rapidly updating agent policies. This mismatch can momentarily destabilize training and reflects a known limitation of the decentralized world model approach, even when using prioritized replay buffers. Nonetheless, MATWM's ability to handle image-based inputs, unlike prior multi-

agent world models, broadens its applicability and represents a significant advancement toward general-purpose, sample efficient multi-agent learning.

4.3. Ablation Study

To evaluate the contribution of key architectural components in MATWM, we perform an ablation study on three core mechanisms: the teammate predictor, prioritized experience replay (PER), and action scaling. For each component, we disable the respective mechanism and measure the performance drop across a representative subset of SMAC maps and image-based environments. The results are summarized in Table 5.

Teammate Predictor. Disabling the teammate predictor consistently leads to substantial performance degradation, particularly in environments involving many agents and requiring coordinated behavior (e.g., 8m, so_many_baneling), where performance often collapses to zero. In smaller-scale or loosely coupled scenarios (e.g., 3m, Pure Coordination in the matrix: Repeated), the model without the predictor still performs reasonably, though it remains outperformed by the full model. In the case of Cooperative Pong, we observe that the teammate predictor provides little to no benefit, and in some runs, even hinders performance. We hypothesize that this is because coordination is not critical in this environment; the primary focus is on tracking the ball, and the actions of the other agent during gameplay may introduce noise or distraction rather than useful information. However, this study confirms that explicitly modeling teammates' actions significantly enhances cooperation in tightly coupled environments.

Prioritized Experience Replay. Replacing PER with uniform sampling results in a noticeable decline in performance, particularly in environments where agents take longer to learn a winning strategy, such as 8m. In such settings, agent behaviors converge more slowly, meaning the replay buffer contains a greater proportion of outdated or suboptimal experiences. PER mitigates this by sampling transitions that are more recent, which allow the world model to track and update with agents' changing behavior. This ensures that updates focus on more relevant experiences, improving imagination and thus sample efficiency. In contrast, in simpler environments like 3m, where agents quickly discover successful strategies and thus stop rapidly changing behavior, the replay buffer is filled with relevant experiences early on, reducing the relative benefit of prioritization.

Action Scaling. Removing the action scaling mechanism also impacts performance, though to a lesser extent than the other ablations. The degradation is most evident in image-based environments, such as Pistonball and Externality Mushrooms: Dense, which can involve multiple agents being present on the screen simultaneously, making it crucial for the world model to identify which agent is conducting the action. We hypothesize that action scaling is particularly beneficial in fully observable settings, where the observation includes the entire environment state and agent differentiation is not implicitly encoded. In contrast, in partially observable environments like SMAC, the focal agent's identity is often loosely embedded within its localized observation as the focal agent's information is represented by the first values in the vector, allowing the model to maintain some degree of agent distinction even without scaling. In environments like Cooperative Pong, however,

where the observation encompasses the whole board, scaling the action distribution helps the model maintain distinct agent behaviors, improving the learning stability of the world model.

Together, these results highlight that MATWM's performance emerges not solely from its predictive world model, but from the synergy between multiple design decisions—including teammate prediction, prioritized sampling, and consistent action scaling.

Table 5: Ablation Study: Impact of Component Removal on Performance

Environment	**Full MATWM**	**– Teammate Predictor**	**– PER**	**– Action Scaling**
3m	83.0	65.0	74.0	**84.0**
8m	**65.0**	0.0	52.0	63.0
so_many_baneling	**74.0**	0.0	59.0	70.0
Cooperative Pong	54.5	**56.3**	52.1	38.4
Pistonball	**92.6**	90.3	85.1	88.4
Externality Mushrooms: Dense	**146.8**	130.0	133.5	135.7
Pure Coordination in the matrix: Repeated	**6.8**	5.9	6.0	6.5

5. Conclusion

We introduce MATWM, a transformer-based multi-agent world model that builds upon STORM, a leading world model architecture for single-agent systems, which achieves strong performance due to its aggregation of many state-of-the-art techniques and architectural advances. MATWM uses imagination-based training, a decentralized world model architecture, and a novel teammate predictor to facilitate coordination in partially observable environments. We also incorporated several elements from state-of-the-art single-agent world models that have not yet been applied to multi-agent world models, such as symlog two-hot rewards, KL balance and free bits, and percentile return normalization. MATWM achieves state-of-the-art performance across a diverse suite of benchmarks, including SMAC, PettingZoo, and MeltingPot, demonstrating strong sample efficiency in complex multi-agent settings by obtaining near-optimal performance in many of these benchmarks in as few as 50K environment interactions, the smallest budget attempted in existing MARL literature. Our experiments highlight the benefits of architectural choices such as prioritized experience replay, action scaling for agent disambiguation, and the explicit modeling of teammate behavior. Notably, we show that MATWM excels not only in traditional vector-based benchmarks but also in complex image-based environments, whereas previous multi-agent world models were limited to vector-based environments.

We hope MATWM provides a solid foundation for future research into more advanced architectures capable of addressing increasingly complex challenges in multi-agent reinforcement learning. In future work, we aim to enhance our framework by investigating additional mechanisms for facilitating multi-agent cooperation, extending support to competitive and mixed cooperative-competitive MARL settings, and addressing limitations in decentralized world models that are related to non-stationarity.

Declaration of Generative AI and AI-Assisted Technologies in the Writing Process

During the preparation of this work, the author(s) used ChatGPT (OpenAI) in order to support language editing. After using this tool, the author(s) reviewed and edited the content as needed and take(s) full responsibility for the content of the publication.

References

[1] J. Foerster, G. Farquhar, T. Afouras, N. Nardelli, S. Whiteson, Counterfactual multi-agent policy gradients, Proceedings of the AAAI Conference on Artificial Intelligence 32 (1) (2018). doi:10.1609/aaai.v32i1.11794.
URL https://ojs.aaai.org/index.php/AAAI/article/view/11794

[2] R. Lowe, Y. Wu, A. Tamar, J. Harb, P. Abbeel, I. Mordatch, Multi-agent actor-critic for mixed cooperative-competitive environments, in: Proceedings of the 31st International Conference on Neural Information Processing Systems (NeurIPS), NeurIPS '17, Curran Associates Inc., Red Hook, NY, USA, 2017, pp. 6382–6393.

[3] C. Yu, A. Velu, E. Vinitsky, J. Gao, Y. Wang, A. Bayen, Y. Wu, The surprising effectiveness of ppo in cooperative multi-agent games, in: Proceedings of the 36th International Conference on Neural Information Processing Systems (NeurIPS), NeurIPS '22', Curran Associates Inc., Red Hook, NY, USA, 2022, p. 1787.

[4] T. Rashid, M. Samvelyan, C. Schroeder de Witt, G. Farquhar, J. Foerster, S. Whiteson, Monotonic value function factorisation for deep multi-agent reinforcement learning, Journal of Machine Learning Research 21 (1) (2020) 178:1–178:51.

[5] J. Wang, Z. Ren, T. Liu, Y. Yu, C. Zhang, Qplex: Duplex dueling multi-agent q-learning, in: Proceedings of the 9th International Conference on Learning Representations (ICLR), 2021.
URL https://openreview.net/forum?id=Rcmk0xxIQV

[6] K. Son, D. Kim, W. J. Kang, D. E. Hostallero, Y. Yi, Qtran: Learning to factorize with transformation for cooperative multi-agent reinforcement learning, in: Proceedings of the 36th International Conference on Machine Learning (ICML), PMLR, 2019, pp. 5887–5896.

[7] P. Sunehag, G. Lever, A. Gruslys, W. M. Czarnecki, V. Zambaldi, M. Jaderberg, M. Lanctot, N. Sonnerat, J. Z. Leibo, K. Tuyls, T. Graepel, Value-decomposition networks for cooperative multi-agent learning based on team reward, in: Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS), AAMAS '18, International Foundation for Autonomous Agents and Multiagent Systems, Stockholm, Sweden, 2018, pp. 2085–2087.

[8] P. Hernandez-Leal, B. Kartal, M. E. Taylor, A very condensed survey and critique of multiagent deep reinforcement learning, in: Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS), AAMAS '20, International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 2020, pp. 2146–2148.

[9] J. Ding, Y. Zhang, Y. Shang, Y. Zhang, Z. Zong, J. Feng, Y. Yuan, H. Su, N. Li, N. Sukiennik, F. Xu, Y. Li, Understanding world or predicting future? a comprehensive

survey of world models (2024). arXiv:2411.14499.
URL https://arxiv.org/abs/2411.14499

[10] Z. Xu, B. Zhang, Y. Zhan, Y. Baiia, G. Fan, et al., Mingling foresight with imagination: Model-based cooperative multi-agent reinforcement learning, in: Advances in Neural Information Processing Systems, Vol. 35, 2022, pp. 11327–11340.

[11] V. Egorov, A. Shpilman, Scalable multi-agent model-based reinforcement learning, in: Proceedings of the 21st International Conference on Autonomous Agents and Multiagent Systems (AAMAS), AAMAS '22, International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, 2022, pp. 381–390.

[12] Y. Zhang, C. Bai, B. Zhao, J. Yan, X. Li, X. Li, Decentralized transformers with centralized aggregation are sample-efficient multi-agent world models (2024). arXiv:2406.15836.
URL https://arxiv.org/abs/2406.15836

[13] W. H. Guss, B. Houghton, N. Topin, A. Velu, S. Codel, M. Alfredo, S. T. Iqbal, D. Dey, S. Wong, K. Gopalakrishnan, et al., Minerl: A large-scale dataset of minecraft demonstrations, in: Proceedings of the 28th International Joint Conference on Artificial Intelligence (IJCAI), 2019.

[14] W. Zhang, G. Wang, J. Sun, Y. Yuan, G. Huang, Storm: Efficient stochastic transformer based world models for reinforcement learning, Advances in Neural Information Processing Systems 36 (2023) 27147–27166.

[15] D. Hafner, T. Lillicrap, M. Norouzi, J. Ba, Mastering atari with discrete world models (2020). arXiv:2010.02193.
URL https://arxiv.org/abs/2010.02193

[16] D. Hafner, J. Pasukonis, J. Ba, T. Lillicrap, Mastering diverse domains through world models (2024). arXiv:2301.04104.
URL https://arxiv.org/abs/2301.04104

[17] J. Robine, M. Höftmann, T. Uelwer, S. Harmeling, Transformer-based world models are happy with 100k interactions, in: Proceedings of the Eleventh International Conference on Learning Representations (ICLR), 2023.
URL https://openreview.net/forum?id=TdBaDGCpjly

[18] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, I. Polosukhin, Attention is all you need, in: Advances in Neural Information Processing Systems, Vol. 30, Curran Associates, Inc., 2017, pp. 5998–6008.
URL https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf

[19] D. P. Kingma, T. Salimans, R. Jozefowicz, X. Chen, I. Sutskever, M. Welling, Improved variational inference with inverse autoregressive flow, Advances in neural information processing systems 29 (2016).

[20] T. Feng, W. Wang, Y. Yang, A survey of world models for autonomous driving (2025). arXiv:2501.11260.
URL https://arxiv.org/abs/2501.11260

[21] F.-M. Luo, T. Xu, H. Lai, X.-H. Chen, W. Zhang, Y. Yu, A survey on model-based reinforcement learning, Science China Information Sciences 67 (2) (2024) 121101.

[22] J. I. Kim, Y. J. Lee, J. Heo, J. Park, J. Kim, S. R. Lim, J. Jeong, S. B. Kim, Sample-efficient multi-agent reinforcement learning with masked reconstruction, PLOS ONE 18 (9) (2023) e0291545.

[23] X. Liu, Y. Chen, H. Li, D. Zhao, Learning future representation with synthetic observations for sample-efficient reinforcement learning, Science China Information Sciences 68 (5) (2025) 150202.

[24] G. Ma, L. Zhang, H. Wang, L. Li, Z. Wang, Z. Wang, L. Shen, X. Wang, D. Tao, Learning better with less: Effective augmentation for sample-efficient visual reinforcement learning, in: Proceedings of the Thirty-Seventh Conference on Neural Information Processing Systems (NeurIPS), 2023.
URL https://openreview.net/forum?id=jRL6ErxMVB

[25] S. Liu, X. S. Zhang, Y. Li, Y. Zhang, J. Cheng, On the data-efficiency with contrastive image transformation in reinforcement learning, in: The Eleventh International Conference on Learning Representations, 2023.

[26] D. Ha, J. Schmidhuber, World models, Zenodo (2018). doi:10.5281/ZENODO.1207631.
URL https://zenodo.org/record/1207631

[27] D. E. Rumelhart, G. E. Hinton, R. J. Williams, Learning internal representations by error propagation, in: D. E. Rumelhart, J. L. McClelland, the PDP Research Group (Eds.), Parallel Distributed Processing: Explorations in the Microstructure of Cognition. Volume 1: Foundations, MIT Press, Cambridge, MA, USA, 1986, pp. 318–362.
URL https://api.semanticscholar.org/CorpusID:62245742

[28] L. Kaiser, M. Babaeizadeh, P. Milos, B. Osinski, R. H. Campbell, K. Czechowski, D. Erhan, C. Finn, P. Kozakowski, S. Levine, A. Mohiuddin, R. Sepassi, G. Tucker, H. Michalewski, Model-based reinforcement learning for atari, in: Proceedings of the 8th International Conference on Learning Representations (ICLR), 2020.
URL https://openreview.net/forum?id=S1xCPJHtDB

[29] D. Hafner, T. Lillicrap, J. Ba, M. Norouzi, Dream to control: Learning behaviors by latent imagination (2019). arXiv:1912.01603.
URL https://arxiv.org/abs/1912.01603

[30] V. Micheli, E. Alonso, F. Fleuret, Transformers are sample-efficient world models, in: Proceedings of the Eleventh International Conference on Learning Representations (ICLR), 2023.
URL https://openreview.net/forum?id=vhFu1Acb0xb

[31] C. Chen, J. Yoon, Y.-F. Wu, S. Ahn, Transdreamer: Reinforcement learning with transformer world models, arXiv preprint, under review at ICLR (2022).
URL https://openreview.net/forum?id=s3K0arSRl4d

[32] K. Cho, B. van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, Y. Bengio, Learning phrase representations using RNN encoder–decoder for statistical machine translation, in: A. Moschitti, B. Pang, W. Daelemans (Eds.), Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), Association for Computational Linguistics, Doha, Qatar, 2014, pp. 1724–1734. doi:10.3115/v1/D14-1179.
URL https://aclanthology.org/D14-1179/

[33] Q. Liu, J. Ye, X. Ma, J. Yang, B. Liang, C. Zhang, Efficient multi-agent reinforcement learning by planning, in: Proceedings of the Twelfth International Conference on Learning Representations (ICLR), 2024.
URL https://openreview.net/forum?id=CpnKq3UJwp

[34] J. Schrittwieser, I. Antonoglou, T. Hubert, K. Simonyan, L. Sifre, S. Schmitt, A. Guez, E. Lockhart, D. Hassabis, T. Graepel, T. Lillicrap, D. Silver, Mastering atari, go, chess and shogi by planning with a learned model, Nature 588 (2020) 604–609. doi:10.1038/s41586-020-03051-4.

[35] A. Jaegle, F. Gimeno, A. Brock, O. Vinyals, A. Zisserman, J. Carreira, Perceiver: General perception with iterative attention, in: M. Meila, T. Zhang (Eds.), Proceedings of the 38th International Conference on Machine Learning, Vol. 139 of Proceedings of Machine Learning Research, PMLR, 2021, pp. 4651–4664.

[36] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, O. Klimov, Proximal policy optimization algorithms (2017). arXiv:1707.06347.
URL https://arxiv.org/abs/1707.06347

[37] C. Kaplanis, M. Shanahan, C. Clopath, Policy consolidation for continual reinforcement learning (2019). arXiv:1902.00255.
URL https://arxiv.org/abs/1902.00255

[38] W. Ye, S. Liu, T. Kurutach, P. Abbeel, Y. Gao, Mastering atari games with limited data, in: Advances in Neural Information Processing Systems, Vol. 34, Curran Associates, Inc., 2021, pp. 25476–25488.
URL https://proceedings.neurips.cc/paper_files/paper/2021/file/d5eca8dc3820cad9fe56a3bafda65ca1-Paper.pdf

[39] D. P. Kingma, M. Welling, Auto-encoding variational bayes (2013). arXiv:1312.6114.
URL https://arxiv.org/abs/1312.6114

[40] A. van den Oord, O. Vinyals, K. Kavukcuoglu, Neural discrete representation learning, in: Proceedings of the 31st International Conference on Neural Information Processing Systems (NeurIPS), NeurIPS'17, Curran Associates Inc., Long Beach, California, USA, 2017, pp. 6309–6318.

[41] C. Beattie, J. Z. Leibo, D. Teplyashin, T. Ward, M. Wainwright, H. Küttler, A. Lefrancq, S. Green, V. Valdés, A. Sadik, J. Schrittwieser, K. Anderson, S. York, M. Cant, A. Cain, A. Bolton, S. Gaffney, H. King, D. Hassabis, S. Legg, S. Petersen, Deepmind lab (2016). arXiv:1612.03801.
URL https://arxiv.org/abs/1612.03801

[42] E. J. Meyer, A. White, M. C. Machado, Harnessing discrete representations for continual reinforcement learning, Reinforcement Learning Journal 2 (2024) 606–628.

[43] E. Jang, S. Gu, B. Poole, Categorical reparameterization with gumbel-softmax, in: Proceedings of the 5th International Conference on Learning Representations (ICLR), 2017.
URL https://arxiv.org/abs/1611.01144

[44] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, L. D. Jackel, Backpropagation applied to handwritten zip code recognition, Neural Computation 1 (4) (1989) 541–551. doi:10.1162/neco.1989.1.4.541.
URL https://doi.org/10.1162/neco.1989.1.4.541

[45] Y. Bengio, N. Léonard, A. Courville, Estimating or propagating gradients through stochastic neurons for conditional computation (2013). arXiv:1308.3432.
URL https://arxiv.org/abs/1308.3432

[46] R. S. Sutton, A. G. Barto, Reinforcement Learning: An Introduction, MIT Press, Cambridge, MA, USA, 1998.

[47] M. Samvelyan, T. Rashid, C. Schroeder de Witt, G. Farquhar, N. Nardelli, T. G. J. Rudner, C.-M. Hung, P. H. S. Torr, J. Foerster, S. Whiteson, The starcraft multi-agent challenge, in: Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems (AAMAS), AAMAS '19, International Foundation for Autonomous Agents and Multiagent Systems, Montreal, QC, Canada, 2019, pp. 2186–2188.

[48] J. K. Terry, B. Black, N. Grammel, M. Jayakumar, A. Hari, R. Sullivan, L. Santos, R. Perez, C. Horsch, C. Dieffendahl, N. L. Williams, Y. Lokesh, P. Ravi, Pettingzoo: A standard api for multi-agent reinforcement learning, in: Proceedings of the 35th International Conference on Neural Information Processing Systems (NeurIPS), NeurIPS '21, Curran Associates Inc., Red Hook, NY, USA, 2021, pp. 1152–1163.

[49] J. P. Agapiou, A. S. Vezhnevets, E. A. Duéñez-Guzmán, J. Matyas, Y. Mao, P. Sunehag, R. Köster, U. Madhushani, K. Kopparapu, R. Comanescu, D. Strouse, M. B. Johanson, S. Singh, J. Haas, I. Mordatch, D. Mobbs, J. Z. Leibo, Melting pot 2.0 (2023). arXiv:2211.13746.
URL https://arxiv.org/abs/2211.13746

[50] G. Papoudakis, F. Christianos, L. Schäfer, S. V. Albrecht, Benchmarking multi-agent deep reinforcement learning algorithms in cooperative tasks, in: Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks (NeurIPS), 2021.
URL https://arxiv.org/abs/2006.07869

[51] G. Papadopoulos, A. Kontogiannis, F. Papadopoulou, C. Poulianou, I. Koumentis, G. Vouros, An extended benchmarking of multi-agent reinforcement learning algorithms in complex fully cooperative tasks (2025). arXiv:2502.04773.
URL https://arxiv.org/abs/2502.04773

[52] C. Formanek, A. Jeewa, J. Shock, A. Pretorius, Off-the-grid marl: Datasets and baselines for offline multi-agent reinforcement learning, in: Proceedings of the 2023 International Conference on Autonomous Agents and Multiagent Systems (AAMAS), AAMAS '23, International Foundation for Autonomous Agents and Multiagent Systems, London, United Kingdom, 2023, pp. 2442–2444.

Appendix A. Imagination Algorithm

Algorithm 1: Imagined Rollouts with Teammate Prediction

Input: Agents $\mathcal{N}$, world model WM, context observations/actions, context length L, batch size B, imagination rollout length H

Output: Imagined latent states, actions, rewards, terminations, teammate actions, action masks

Initialize per-agent rollout buffers for latent states, actions, rewards, terminations, teammate actions, action masks (optional)

`// Use Context as starting point for imagination rollout`

Encode context observations into latent state sequence $z_{0:L}$

for $t = 0$ **to** $L - 1$ **do**

  Predict next latent state, hidden state, reward, and termination using:

    $(\hat{z}_{t+1}, h_{t+1}, \hat{r}_t, \hat{c}_t) \leftarrow WM(z_t, a_t)$

Store z_L and hidden state h_L in respective buffers

`// Begin imagination rollout`

for $t = 0$ **to** $H - 1$ **do**

  foreach *agent* $n_i \in \mathcal{N}$ **do**

    Get latent $z_t^{(i)}$ and hidden $h_t^{(i)}$ slice for agent batch

    Compute predicted teammate action probabilities:

    $\hat{p}_{i,t}^{(a)} \leftarrow \text{TeammatePredictor}(z_t^{(i)})$

    if *action masking enabled* **then**

      Predict action mask $\hat{m}_t \leftarrow \text{MaskPredictor}(z_t^{(i)})$

      Sample action $a_{i,t} \sim \pi(z_t^{(i)}, h_t^{(i)}, \hat{p}_{i,t}^{(a)}, \hat{m}_t)$

      Store $\hat{m}_t$ in respective buffer

    else

      Sample action $a_{i,t} \sim \pi(z_t^{(i)}, h_t^{(i)}, \hat{p}_{i,t}^{(a)})$

    Store $a_{i,t}$ and $\hat{p}_{i,t}^{(a)}$ in respective buffers

  Predict next $(\hat{z}_{t+1}, h_{t+1}, \hat{r}_t, \hat{c}_t) \leftarrow WM(z_t, a_t)$

  Store results in respective buffers

Return buffers for latent states, actions, rewards, terminations, teammate actions, and (optional) masks

Appendix B. Training Algorithm

Algorithm 2: Joint Training of World Model and Agents in MATWM

Input: Environment $\mathcal{E}$, world model WM, actor-critic agents $\{\pi_i\}_{i=1}^{N}$, replay buffers $\{RB_i\}_{i=1}^{N}$

Parameters *Total steps T, train intervals t_{WM}, t_π, batch sizes, context lengths*
Initialize environment $\mathcal{E}$ and per-agent buffers, context queues
for $t = 1$ **to** T **do**
  foreach *agent i* **do**
    if *RB_i is ready* **then**
      Encode recent observations $\mathcal{O}_i$ into latent z_i using WM
      Predict teammate behavior using WM's predictor
      Sample action $a_i \sim \pi_i(z_i, \hat{a}_{-i})$
    else
      Sample a_i randomly or with action masking
  Step environment $\mathcal{E}$ with $\{a_i\}$, collect next obs, rewards, dones
  Store transitions in each RB_i with teammate action info
  if *episode terminates* **then**
    Reset environment and context buffers
  if $t \bmod t_{WM} = 0$ *and all RB_i ready* **then**
    Sample batches from all RB_i
    Train WM on combined batch (obs, actions, rewards, terminations, teammate actions)
  if $t \bmod t_\pi = 0$ *and all RB_i ready* **then**
    Use WM to imagine trajectories for each agent
    Train each policy π_i using imagined data (rewards, terminations, predicted teammates)

Appendix C. Reproducibility

Code for MATWM is available on GitHub [2]. For all SMAC experiments, we use Starcraft II version SC2.4.1.2.60604. Please note that using different versions may yield slightly different results. Hyperparameter settings for MATWM are shown in Table C.6. The settings for all baselines are taken from [12] for the SMAC experiments and [51] for the PettingZoo and MeltingPot experiments.

[2] github.com/azaddeihim/matwm

[3] These settings are for environments with three or less agents. For environments with four to six agents and for 2c_vs_64z, we use a batch size of 768 with an imagination horizon of 12. For environments with seven or more agents, we use a batch size of 1024 with an imagination horizon of 8.

Table C.6: MATWM Hyperparameter Settings

Hyperparameter	**Value**
Max sequence length	64
Hidden dimension	512
Number of layers	2
Number of attention heads	8
Latent dimension size	32
Number of categories per latent	32
World model train batch length	64
World model train batch size	16
World Model Learning rate	3×10^{-5}
Actor+Critic Learning rate	3×10^{-4}
Optimizer	Adam
Gradient clipping agent	100.0
Gradient clipping world model	1000.0
Replay buffer size	50,000
Replay sampling priority decay	0.9998
KL loss weight (β_1)	0.5
Representation loss weight (β_2)	0.1
Imagination horizon	$16^{\ 3}$
Agent train batch size	512^3
Imagination context length	8
Train world model every n steps	1
Train agent every n steps	1
MLP Encoder Hidden dim	512
MLP Encoder Hidden layers	3
CNN Encoder Kernel Size	4
CNN Encoder Stride	2
CNN Encoder Layers	4