Action-Conditioned Transformers for Decentralized Multi-Agent World Models

Anonymous authors
Paper under double-blind review

Abstract

Multi-agent reinforcement learning (MARL) has achieved strong results on large-scale decision making, yet most methods are model-free, limiting sample efficiency and making coordination harder as teammates' policies evolve during training. Model-based reinforcement learning (MBRL) can reduce data usage, but planning and search scale poorly with joint action spaces. We adopt a world model approach to long-horizon coordination while avoiding expensive planning. We introduce MACT, a decentralized transformer world model with linear complexity in the number of agents. Each agent processes discretized observation–action tokens with a shared transformer, while a single cross-agent Perceiver step provides global context under centralized training and decentralized execution. MACT targets long-horizon coordination by coupling Perceiver-derived global context with an action-conditioned contrastive objective that predicts future latent representations over a short horizon conditioned on planned actions. Experiments on the StarCraft Multi-Agent Challenge (SMAC) under tight data budgets show that MACT is competitive with strong model-free baselines and prior world-model variants, with larger gains on coordination-heavy scenarios.

1 Introduction

Model free multi-agent algorithms such as QMIX Rashid et al. (2020), QPLEX Wang et al. (2021), and MAPPO Yu et al. (2022) can achieve robust long term returns, but they do so at the cost of millions of environment interactions. Two structural factors drive this sample hunger: the exponential growth of the joint observation action space as team size increases Liu et al. (2024) and the non stationarity that emerges when each agent's data distribution changes in response to its teammates' evolving policies Gronauer & Diepold (2022). For example, consider a 'focus fire' movement on the StarCraft multi-agent Challenge Samvelyan et al. (2019) (SMAC) environments, where a group of units must be commanded to attack a single enemy to eliminate it faster. Success depends on understanding the delayed consequences of the team's joint actions. This is the type of long horizon reasoning that models trained on one step prediction have a hard time grasping. In single agent settings, model based reinforcement learning (MBRL) addresses similar issues by training a latent world model Ha & Schmidhuber (2018) (WM) that can be rolled forward in imagination, thereby replacing expensive real transitions with synthetic ones, like Dreamer Hafner et al. (2020). The Dreamer transformer based successor TWISTER Burchi & Timofte (2025), and the domain robust DreamerV3 Hafner et al. (2025) show that accurate latent dynamics can cut sample cost by an order of magnitude when the objective encourages multi step predictive structure.

Transferring this promise to multi-agent scenarios has proved difficult. MAMBA Egorov & Shpilman (2022) swapped Dreamer's recurrent core for an agent aware LSTM, yet all agents still shared the same latent state, so the model scaled poorly beyond a handful of entities. A more scalable design arrived with MARIE Zhang et al. (2025), which assigns each agent its own transformer for local token dynamics and injects joint context through a lightweight Perceiver cross attention layer. However, both MAMBA and MARIE supervise their models with one step token reconstruction losses, which can bias representations toward short-term correlations, making long-horizon coordination more challenging.

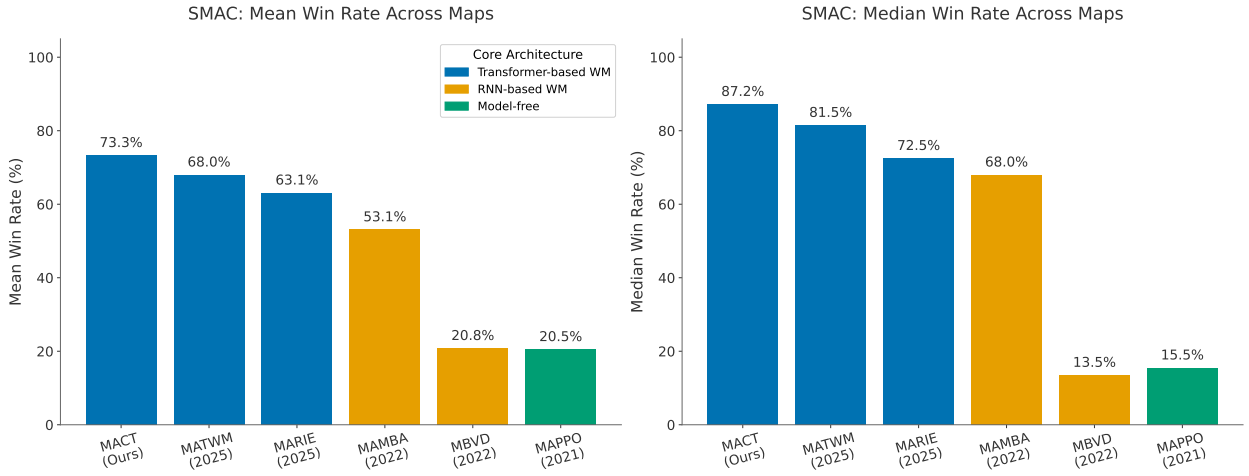


Figure 1: Mean (left) and median (right) win rates across SMAC maps. Bars are color-coded by their used learning methodology.

TWISTER showed in the single agent domain that a transformer equipped with action-conditioned contrastive predictive coding (AC-CPC) can exploit its full representational capacity. Instead of only reconstructing the next latent state, the model predicts a sequence of future latents $\{z_{t+1}, \ldots, z_{t+K_{\text{cpc}}}\}$ conditioned on an action window $\{a_t, \ldots, a_{t+K_{\text{cpc}}-1}\}$ taken from the sampled replay segment and learns by contrasting the true future against negative samples in the batch. This long-horizon objective encourages temporally predictive abstractions over multiple steps, and TWISTER reports that AC-CPC improves performance in the Atari-100k regime relative to one-step objectives.

Our goal is to bring the benefits of model-based learning to cooperative control under tight data budgets without sacrificing simplicity or scale. We present **MACT**, a **M**ulti-agent **A**ction-**C**onditioned **T**ransformer that predicts several future latent states given a short planned action window. Each agent is processed locally by a shared transformer, and a single Perceiver pass supplies light team context, which keeps computation near linear in team size. While MACT draws inspiration from MARIE's scalable centralized-training and decentralized-execution (CTDE) architecture and TWISTER's action-conditioned CPC objective, their combination is not plug-and-play in multi-agent settings: one must decide *what* representation to predict under CTDE and *which* actions should condition that prediction, while avoiding trivial token matching and preserving scalable cross-agent context. MACT resolves these choices by predicting future team-conditioned Perceiver latents from each agent's local latent and a short per-agent action window, using an augmented future view as the positive target. Figure 1 previews the results: across SMAC maps, world model approaches including MACT increase mean win rate under tight data budgets. In this paper we describe the objective and training procedure, evaluate on SMAC with ablations on horizon length and context, and analyze the coordination patterns that emerge.

Our contributions are described as follow:

- We design an action-conditioned, multi-step contrastive learning objective for decentralized MARL by resolving CTDE-specific action-conditioning and target choices: each agent predicts future team-conditioned Perceiver latents from its local latent and a short per-agent action window, using an augmented future view as positives to avoid trivial matching.
- On SMAC under tight data budgets, MACT is competitive with strong model-free and prior world-model baselines, with larger gains on coordination-heavy maps.
- Ablations show that moderate prediction horizons and light observation augmentation help, and that per-agent conditioning consistently outperforms team-aggregated conditioning.

2 Related Work

Model-free MARL has progressed through value-factorization methods (VDN Sunehag et al. (2018), QMIX Rashid et al. (2020), QPLEX Wang et al. (2021)) and policy-gradient variants (MAPPO Yu et al. (2022), HAPPO Kuba et al. (2022)). All of them follow the CTDE recipe: global information is used during learning, but each agent runs a local policy at test time. Because every joint configuration must still be sampled, their data budgets remain in the millions. This is an obstacle that our world model approach seeks to overcome.

Single-agent MBRL pre-trains a generative model of environment dynamics and then improves a policy inside that model. Early versions such as SimPLe Kaiser et al. (2019) employed LSTMs, while Dreamer switched to a recurrent state-space model and introduced symlog rewards. DreamerV3 Hafner et al. (2025) refined the recipe, achieving domain robustness without per-task tuning. Several groups replaced RNNs with transformers to exploit parallel training: IRIS Micheli et al. (2023) maps each frame to a 4×4 grid of VQ-VAE van den Oord et al. (2017) tokens and processes the result with a spatial-temporal transformer. TWM Robine et al. (2023) concatenates observation, action, and reward tokens and trains a Transformer-XL. STORM Zhang et al. (2023) adds stochastic latent variables to a GPT-like backbone and reports strong human-normalized scores on Atari-100k. These works validate transformers as world model cores but still rely on next-step prediction and therefore do not fully tap long-horizon capacity.

Contrastive objectives address this limitation. CPC van den Oord et al. (2018) maximizes mutual information between present and future representations by contrasting the true future against negatives. In visual Reinforcement Learning (RL), the approach of using contrastive learning in combination with RL was popularized by the method CURL Laskin et al. (2020), which treats different data-augmented views of the same observation as a positive pair to learn spatial features but do not pay attention to the temporal and action-conditioned nature of control tasks. Building on temporal and action-driven features, TACO Zheng et al. (2023) introduces a temporal, action-driven contrastive loss designed to predict the future. TACO maximizes mutual information between a current state paired with a future action sequence and the resulting future state, which allows TACO to learn both state and action representations. A similar principle that applies in AC-CPC, which also includes the planned action sequence, removing ambiguities in passive video prediction. TWISTER Burchi & Timofte (2025) is the first to pair AC-CPC with a transformer world model, showing that long-horizon objectives unlock transformer capacity and surpass RNN baselines in low-data regimes.

Multi-agent world models face an additional scalability challenge: the joint observation–action space grows exponentially with team size. MAMBA Egorov & Shpilman (2022) adapts Dreamer to SMAC but keeps a single shared latent state, which limits scalability. MARIE Zhang et al. (2025) distributes token dynamics over agent-specific transformers and injects global context through one step of Perceiver cross-attention. Architecturally, MACT inherits MARIE's near-linear scaling in team size through per-agent local attention and a single Perceiver cross-attention step (formal complexity in Appendix B). Other methods have explored augmenting the transformer world models with a teammate predictor module (MATWM) Deihim et al. (2025) or using learned models from value-decomposition data methods Xu et al. (2022). However, a common weakness in this prior works is the shallow one-step reconstruction loss, which causes rollouts to drift after a few steps, especially when rewards depend on coordinated actions spread over time.

3 Methodology

In this section we first describe the MACT world-model architecture (Subsections 3.1–3.3), then the training objectives applied to that architecture (Subsections 3.4–3.6), and finally the imagination-based actor–critic procedure that consumes the learned world model (Subsection 3.7). This organization makes explicit how latent states produced by the architecture flow into the loss and, in turn, into the control pipeline.

On SMAC environments, each map is modeled as a Dec-POMDP $\langle \mathcal{S}, \mathcal{A}^{1:N}, P, R, \Omega^{1:N}, \gamma \rangle$. At time t every allied unit $i \in \{1{:}N\}$ receives a feature vector $o_t^i \in \mathbb{R}^{d_o}$ containing its own hit-points, cool-down, terrain height, relative distances to the nearest enemies and allies, and boolean flags. This boolean flags can be for example "enemy in range". The raw dimensionality is modest, around $d_o \approx 70$ for map `3s_vs_5z`, but the joint observation space $\Omega^{1:N} = \Omega^1 \times \ldots \times \Omega^N$ still grows exponentially with N. The agent then chooses

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

Aspect	**MACT (Ours)**	**MARIE**	**MATWM**	**MAMBA**	**MBVD**
Backbone Architecture	Transformer	Transformer	Transformer	GRU	GRU
Latent Representation	VQ-VAE	VQ-VAE	Categorical VAE	Categorical VAE	VAE
Critic Type	Centralized	Centralized	Semi-centralized	Centralized	Centralized
World Model Type	Hybrid	Hybrid	Decentralized	Centralized	Centralized
Agent Training Strategy	PPO-style	PPO-style	DreamerV3-style	PPO-style	Deep Q-learning
Prediction Horizon	K_{cpc}=8 steps	Next state	Next state	Next state	Next state

a discrete action $a_t^i \in \mathcal{A}^i$, where $\mathcal{A}^i$ is a finite categorical action set (move-direction, attack-enemy, stop, etc.). On SMAC we assume a shared discrete action set across agents, i.e., $\mathcal{A}^i = \mathcal{A}$ for all $i \in \{1{:}N\}$, and thus $|\mathcal{A}^i| = |\mathcal{A}|$. We index actions as integers $a_t^i \in \{1{:}|\mathcal{A}|\}$, and use the corresponding one-hot vectors $\text{onehot}(a_t^i) \in \{0,1\}^{|\mathcal{A}|}$ only when their dimensionality is relevant, such as in the action-conditioned CPC objective. Executing the joint action $a_t^{1:N}$ through the unknown kernel P yields the next state s_{t+1} and the shared reward r_t. An episode ends when one army is eliminated or a time-limit is reached. The goal is to maximize the discounted return $\mathbb{E}[\sum_{t=0}^{\infty} \gamma^t r_t]$ while respecting decentralized execution. We use discount factor $\gamma \in (0,1)$ for return computation, and a continuation indicator c_t for episode termination. An overall view of our method is described in Figure 2.

Design choices relative to prior work. To make clear how MACT relates to existing multi-agent world-model baselines, Table 1 summarizes key architectural and training choices. In particular, MACT is closest in overall pipeline to MARIE (tokenized per-agent dynamics with a Perceiver-style cross-agent context under CTDE), but differs in its prediction objective: we introduce an action-conditioned, multi-step contrastive loss that explicitly trains a K_{cpc}-step predictive representation rather than only next-step reconstruction. Following MARIE, we avoid joint cross-agent self-attention over all agents' tokens (which would scale quadratically with the effective sequence length) and instead use a single Perceiver-style cross-attention step that aggregates the current-step joint token set of length $N(K_{\text{tok}}{+}1)$ into N agent-wise global features (see Appendix B).

Vector-to-token conversion. First, MACT converts each agent's continuous observation vector into a more structured sequence of discrete tokens. To do this, we use a small vector-quantized auto-encoder $(\text{E}, \text{D}, \mathcal{Z})$. This process creates a learned vocabulary for the features of the environment. The encoder takes the full observation vector o_t^i, splits it into $K_{\text{tok}} = 8$ smaller pieces, and for each piece, it finds the best-matching "word", a code-book index $x_{t,j}^i \in \{1{:}256\}$.

$$x_t^i = (x_{t,1}^i, \ldots, x_{t,K_{\text{tok}}}^i) \quad \text{with} \quad x_t^i \in \mathcal{Z}^{K_{\text{tok}}}. \tag{1}$$

Throughout the paper, each $x_{t,j}^i \in \mathcal{Z}$ is a discrete code-book index. We denote the tokenizer code-book as $\mathcal{Z} = \{1, \ldots, 256\}$. Whenever a continuous representation is required, we explicitly apply the shared embedding (Eq. equation 2). That is, we write $e(x_{t,j}^i)$ for the corresponding D_x-dimensional embedding.

Using discrete tokens instead of raw continuous numbers brings two practical advantages: it stabilizes training, because predicting the correct "word" from a fixed 256-entry vocabulary is a standard cross-entropy classification problem that avoids the large, unstable gradients of direct regression. At the same time, it exposes useful compositional structure, since treating observations as a token sequence lets the model learn language-like dependencies. One example is to make connections between a token for "small distance to an enemy" that will be often followed by "enemy in range". Finally, to form the input for a single time step, the K_{tok} observation tokens are concatenated with a single discrete action token (the categorical choice a_t^i) and a placeholder aggregation token $*_t^i$ to yield the step block $\mathbf{X}_t^i = \left[x_t^i,\, a_t^i,\, *_t^i\right]$. We thus count each action as one token in the sequence; its one-hot encoding is only used in contexts where the vector dimensionality matters explicitly (e.g., Eq. equation 5).

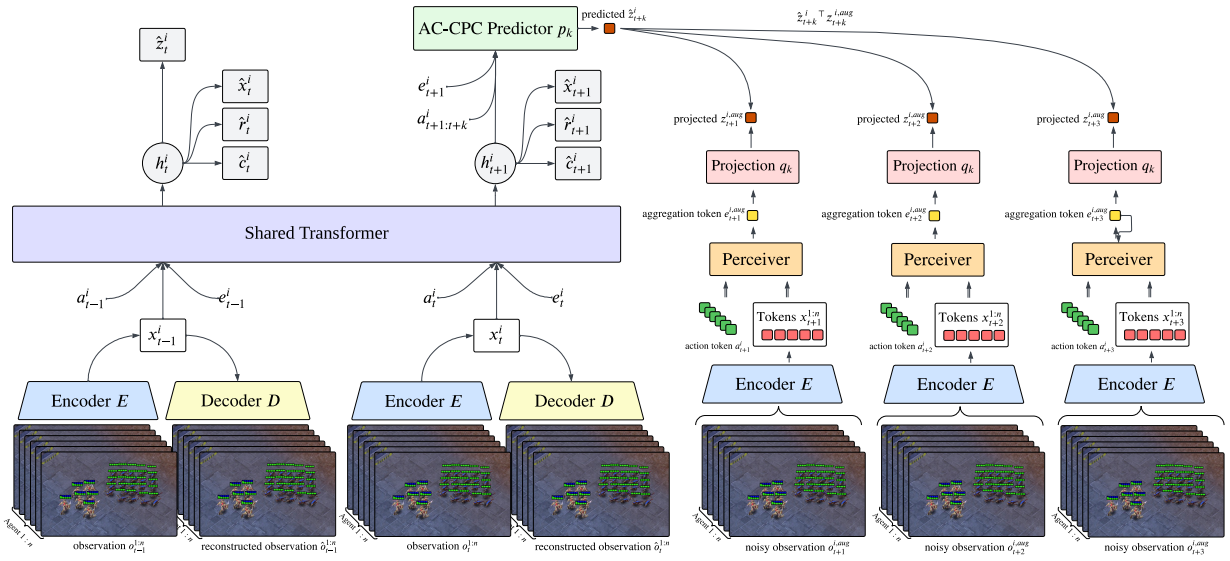


Figure 2: **World-model training in MACT.** Observations $o_t^{1:N}$ are tokenized by a VQ encoder E into discrete codes $x_t^{1:N}$, and a decoder D reconstructs $\hat{o}_t^{1:N}$ to train the tokenizer with a reconstruction loss. Given token histories and executed actions, a shared per-agent Transformer produces local latents h_t^i, from which one-step prediction heads output $\hat{x}_{t+1}^i$, reward $\hat{r}_t^i$, and continuation $\hat{c}_t^i$. For AC-CPC, the predictor consumes the current representation $\left[h_t^i;\, e_t^i\right]$ together with a per-agent action window $a_{t:t+k-1}^i$ and predicts a future representation $\hat{z}_{t+k}^i$. This prediction is contrasted (InfoNCE) against the projected Perceiver target $z_{t+k}^{i,\mathrm{aug}} = q_k(e_{t+k}^{i,\mathrm{aug}})$, where $e_{t+k}^{i,\mathrm{aug}}$ is computed from a dropout/noise-augmented future observation.

Local Transformer dynamics. The transformer process an agent's history and produce a summary of its current situation. To prepare the input, token blocks are taken for each agent i individually from the start of the episode up to the current time $0{:}t$ and flattened into a long sequence.

A block-sparse Transformer ϕ processes this sequence. The block-sparse design is important: it restricts self-attention so that each agent's Transformer only focuses on its own history. In this stage, the transformer cannot see the raw history of other agents. But after reading its entire history, the model produces a single vector that summarizes the current step. This vector is the local summary, $h_t^i \in \mathbb{R}^{D_x}$. This summary is simply the Transformer's output state at the position of the aggregation token. Here D_x is the token-embedding width (256).

Centralized latent aggregation. At each environment step t, we form a token set by concatenating all agents' observation tokens and the current discrete action token:

$$U_t \in \mathbb{R}^{N(K_{\mathrm{tok}}+1)\times D_x}, \tag{2}$$

where K_{tok} is the number of observation tokens per agent per step, and D_x is the Transformer embedding width. We then apply a Perceiver-style cross-attention module that returns one latent per agent:

$$E_t = \mathrm{LN}\Big(Q + \mathrm{softmax}\big(\tfrac{(QW_Q)(U_tW_K)^\top}{\sqrt{d}}\big)\,(U_tW_V)\Big), \tag{3}$$

where $Q \in \mathbb{R}^{N\times D_e}$ are learned per-agent queries, $U_t \in \mathbb{R}^{N(M+1)\times D_x}$ are current-step tokens, $W_Q \in \mathbb{R}^{D_e\times d}$, $W_K \in \mathbb{R}^{D_x\times d}$, and $W_V \in \mathbb{R}^{D_x\times D_e}$ are learned projections, and $E_t = [e_t^1;\ldots;e_t^N] \in \mathbb{R}^{N\times D_e}$, and d is the query/key dimension.

In summary, Perceiver is a standard cross-attention block with learned queries (one per agent) and key/value projections, producing agent-wise latents by attending to the joint token set at the current time step. Thus, each global latent e_t^i mixes information from all agents. Finally, the e_t^i vectors are appended as extra tokens to the next step's Transformer input, propagating global context forward.

Prediction heads and one-step losses. Following MARIE, the prediction heads attach directly to tokens produced by the local transformer. The observation head reads the k-th latent slot, not e_t^i, and outputs a categorical distribution over the code-book to predict $\hat{x}_{t+1,k}^i$ conditioned on $x_{\leq t,\cdot}^i$, $a_{\leq t}^i$, $e_{\leq t}^i$, and the previously generated slots $\hat{x}_{t+1,<k}^i$. This auto-regressive factorization across the K slots captures intra-step structure such as the geometry of nearby units. The reward head maps the aggregation-slot hidden state h_t^i through an MLP to produce a scalar reward prediction $\hat{r}_t^i$, and the discount head predicts the continuation indicator $\hat{c}_t^i \in (0,1)$. Finally, the one-step likelihood objective $\mathcal{L}_{\text{dyn}}$ averages token, reward, and continuation losses over agents and timesteps:

$$\mathcal{L}_{\text{dyn}} = \mathbb{E}_{i,t}\Big[\sum_{k=1}^{K_{\text{tok}}} \text{CE}\big(\hat{x}_{t+1,k}^i, x_{t+1,k}^i\big) + \text{SmoothL1}\big(\hat{r}_t^i,\, \text{symlog}(r_t)\big) + \text{BCE}\big(\hat{c}_t^i, c_t\big)\Big] \tag{4}$$

where the expectation denotes averaging over the sampled replay segment.

The previous components fully specify the latent world model: given the tokenized observation–action histories, the shared Transformer plus the single Perceiver step produce per-agent local latents h_t^i as well as global context features e_t^i. The training objectives introduced in the next subsections (the one-step likelihood $\mathcal{L}_{\text{dyn}}$, the action-conditioned CPC loss $\mathcal{L}_{\text{cpc}}$, and the auxiliary availability term) are applied to these latent quantities during world-model training. In the imagination-based actor–critic phase (Paragraph 3), we follow MARIE and decode the world model's predicted observation tokens to reconstructed observations $\hat{o}_t^i$, which serve as inputs to the decentralized actors. The centralized critic aggregates information across agents (e.g., by concatenating $\{h_t^i\}_{i=1}^N$ or $\{\hat{o}_t^i\}_{i=1}^N$) under CTDE. The Perceiver outputs e_t^i influence control only indirectly through their effect on the learned latent dynamics.

Action-conditioned contrastive prediction (per-agent). Next-step supervision does not force the aggregation state h_t^i to carry information about how *this agent's* planned actions will shape its future when teammates are also moving. Our AC–CPC objective therefore asks each agent to predict, in latent space, what its Perceiver context will be several steps ahead given its own action window. Concretely, for $k \in \{0{:}K_{\text{cpc}}{-}1\}$ we form the context

$$\underbrace{h_t^i}_{\text{Transformer}} \;\|\; \underbrace{e_t^i}_{\text{Perceiver}} \;\|\; \underbrace{a_{t:t+k-1}^i}_{\text{Agent actions}} \quad (\in \mathbb{R}^{D_x+D_e+k\,|\mathcal{A}|}), \tag{5}$$

where $h_t^i \in \mathbb{R}^{D_x}$ is the aggregation-slot hidden state of agent i at time t, $e_t^i \in \mathbb{R}^{D_e}$ is its Perceiver-derived global context, and $a_{t:t+k-1}^i \in \{0,1\}^{k|\mathcal{A}|}$ is the concatenation of the next k one-hot encodings $\text{onehot}(a_t^i), \ldots, \text{onehot}(a_{t+k-1}^i)$, and $|\mathcal{A}|$ is the per-agent discrete action vocabulary. A two-layer MLP $p_k : \mathbb{R}^{D_x+D_e+k|\mathcal{A}|} \to \mathbb{R}^{d_z}$ maps this concatenation to a projected prediction and a two-layer projector $q_k : \mathbb{R}^{D_e} \to \mathbb{R}^{d_z}$ produces the projected target. Let e'^i_{t+k} denote the Perceiver output computed from a feature-dropout augmented observation view at time $t{+}k$. We then define the projected prediction and projected target as:

$$\hat{z}_{t+k}^i = p_k\Big(\big[h_t^i \,\|\, e_t^i \,\|\, a_{t:t+k-1}^i\big]\Big), \quad z_{t+k}^i = q_k\big(e'^i_{t+k}\big). \tag{5.1}$$

To avoid trivial token matching and encourage action-relevant invariance, the target e'^i_{t+k} is computed from a feature-dropout augmentation on vector observations. With a minibatch of Q positives (agent–time pairs), we index pairs by $q \in \{1, \ldots, Q\}$ (each q corresponds to some (i,t)). Let $Z_{t+k} = [z_{t+k}^{(1)}, \ldots, z_{t+k}^{(Q)}] \in \mathbb{R}^{d_z \times Q}$. The InfoNCE term uses dot-product logits $\hat{z}_{t+k}^{(q)\top} Z_{t+k} \in \mathbb{R}^Q$ and cross-entropy over the index of the positive, the diagonal match:

$$\ell_k = \tfrac{1}{Q} \sum_{q=1}^{Q} \text{CE}\big(\hat{z}_{t+k}^{(q)\top} Z_{t+k},\, q\big). \tag{5.2}$$

Intuitively, the predictor must learn a causal association: "if agent i executes $a_{t:t+k-1}^i$ while embedded in team context e_t^i, its future context should look like z_{t+k}^i." This resolves temporal ambiguity (the action window disambiguates which future is correct), mitigates credit assignment by linking an agent's actions to

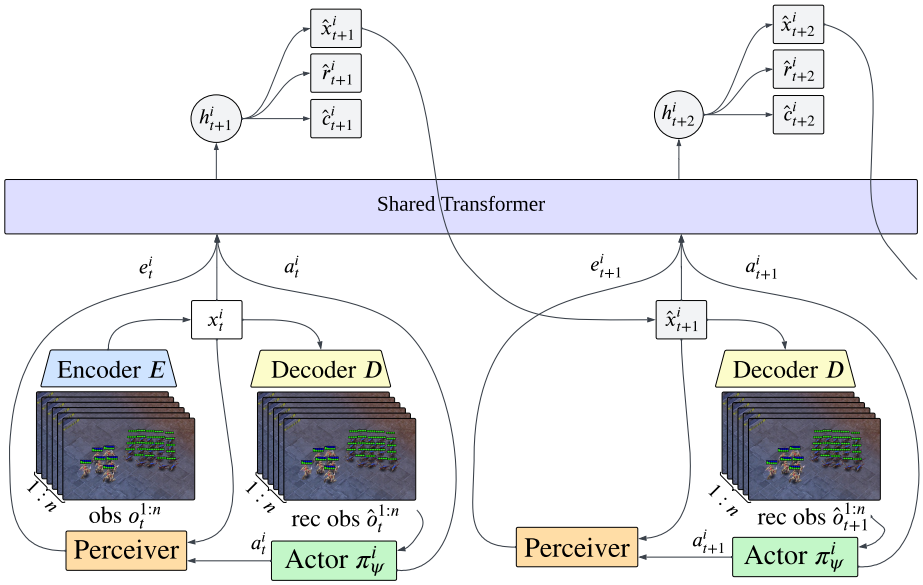


Figure 3: A replay state is encoded into tokens $x_t^{1:N}$ (VQ encoder E) and reconstructed as $\hat{o}_t^{1:N}$ (decoder D). The frozen world model predicts $\hat{x}_{t+1}^{1:N}$, rewards $\hat{r}_t$, and continuation $\hat{c}_t$, which are decoded to $\hat{o}_{t+1}^{1:N}$ and fed back to the actors. Each agent samples actions from reconstructed inputs, $a_t^i \sim \pi_\psi^i(\hat{o}_t^i)$, unrolling for H_{roll} steps without environment interaction.

its own future state, and reduces non-stationarity because teammates' reactions are already absorbed into e_t^i. We include a same-step term (k=0) to align spaces and stabilize training, and weight farther horizons geometrically with λ_{cpc}=0.75:

$$\mathcal{L}_{\text{cpc}} = \sum_{k=0}^{K_{\text{cpc}}-1} \frac{\lambda_{\text{cpc}}^k}{\sum_{j=0}^{K_{\text{cpc}}-1} \lambda_{\text{cpc}}^j} \ell_k, \tag{6}$$

ensuring gradients from distant steps remain present yet do not dominate.

Available-action regularizer. SMAC provides an available-action mask $m_t^i \in \{0,1\}^{|\mathcal{A}|}$ that disables infeasible actions. Without extra care, the logits of masked actions can drift to large negative values, which we found to hinder optimization. Following prior work, we add an auxiliary head that predicts the next-step availability mask m_{t+1}^i from the agent latent, yielding $\hat{m}_{t+1}^i \in (0,1)^{|\mathcal{A}|}$. We train it with a per-action Bernoulli loss

$$\mathcal{L}_{\text{av}} = \mathbb{E}_{i,t}\Big[\text{BCE}\big(\hat{m}_{t+1}^i,\, m_{t+1}^i\big)\Big]. \tag{7}$$

We add this term to the world-model objective alongside the one-step likelihood and AC-CPC:

$$\mathcal{L}_{\text{WM}} = \mathcal{L}_{\text{dyn}} + \beta_{\text{cpc}}\mathcal{L}_{\text{cpc}} + \beta_{\text{av}}\mathcal{L}_{\text{av}}. \tag{8}$$

We found the natural scales of these terms to be comparable and therefore use equal weights ($\beta_{\text{cpc}} = \beta_{\text{av}} = 1$). Training details and hyperparameters are deferred to Appendix B.

Imagination-based actor–critic. After every world model update we keep its parameters fixed and perform imagination rollouts in latent space, following the MARIE training pipeline. Given an initial real observation from the replay buffer, we encode it once with the tokenizer and world model to obtain the per-agent local latents $h_0^{1:N}$. We then unroll $H_{\text{roll}} = H$ latent steps without accessing the environment.

At each imagined step t, every agent runs a decentralized actor $\pi_\psi^i : \mathbb{R}^{d_o} \to \mathbb{R}^{|\mathcal{A}|}$ that takes the reconstructed observation $\hat{o}_t^i$ as input and outputs action logits. We obtain $\hat{o}_t^i$ by decoding the world model's predicted

Table 2: Comparison of methods on SMAC. Values report median (std) win rate (%) over seeds.

Map	Steps	MACT (Ours)		MATWM Deihim et al. (2025)		MARIE Zhang et al. (2025)		MAMBA Egorov & Shpilman (2022)		MBVD Xu et al. (2022)		MAPPO Yu et al. (2022)	
		Med	Std	Med	Std	Med	Std	Med	Std	Med	Std	Med	Std
2m_vs_1z	50K	95.0	0.0	**98.0**	3.2	95.0	4.4	91.0	6.2	41.0	20.7	51.0	10.3
2s_vs_1sc	50K	**98.7**	2.2	96.0	5.7	90.0	9.1	80.0	7.3	0.0	1.2	18.0	7.6
2s3z	50K	65.0	1.7	**80.0**	9.0	71.0	8.6	68.0	12.1	28.0	17.5	13.0	3.0
3m	50K	**96.7**	3.4	83.0	10.4	78.0	14.1	68.0	7.7	60.0	9.2	54.0	6.3
3s_vs_3z	50K	80.0	3.4	**87.0**	19.4	85.0	21.8	77.0	23.7	0.0	0.0	0.0	0.0
3s_vs_4z	50K	4.1	7.2	**12.0**	4.8	0.0	0.8	4.0	1.4	0.0	0.0	0.0	0.0
8m	50K	**95.0**	5.0	67.0	24.9	72.0	7.1	68.0	6.4	52.0	18.9	38.0	4.9
MMM	50K	**39.2**	32.6	7.0	4.7	1.0	1.6	3.0	3.5	0.0	0.0	0.0	0.0
so_many_baneling	50K	**94.4**	7.9	86.0	22.9	73.0	12.4	66.0	14.2	27.0	12.3	31.0	7.6
3s_vs_5z	200K	65.0	11.7	64.0	26.5	**66.0**	28.0	6.0	10.1	0.0	0.0	0.0	0.0
Mean	—	**73.3**	—	68.0	—	63.1	—	53.1	—	20.8	—	20.5	—
Median	—	**87.2**	—	81.5	—	72.5	—	68.0	—	13.5	—	15.5	—

observation tokens (i.e., $\hat{o}_t^i = D(\hat{x}_t^i)$). The actor is implemented as a two-layer MLP with hidden width 256 and ELU activations. Infeasible actions are masked using the SMAC available-action vector, and an action is sampled from the resulting categorical distribution. The frozen world model then uses these imagined actions to predict synthetic rewards, continuation flags, and the next-step observation tokens $\hat{x}_{t+1}^{1:N}$, which are decoded to $\hat{o}_{t+1}^{1:N}$ for the next actor step.

For value estimation we employ a centralized critic $V_\omega : \mathbb{R}^{ND_x} \to \mathbb{R}$ that aggregates information across agents by concatenating the local latents $\{h_t^1, \ldots, h_t^N\}$ and passing them through a two-layer MLP (hidden width 256, ELU). The resulting scalar output approximates the value of the imagined latent state. Using the synthetic rewards and continuation flags, we compute λ-returns along the latent trajectory and update the actors by advantage policy-gradient with an entropy bonus, and the critic by a symlog mean-squared error loss. During this phase gradients do not flow into the world model, ensuring that policy learning does not destabilize the learned dynamics.

4 Experiments

We evaluate MACT on the StarCraft Multi-Agent Challenge (SMAC). The benchmark suite covers both easier micro scenarios (`2m_vs_1z`, `2s_vs_1sc`, `3m`, `8m`, `so_many_baneling`) and maps that demand tighter coordination or longer horizons (`2s3z`, `3s_vs_3z`, `3s_vs_4z`, `MMM`, `3s_vs_5z`). Following the low–data protocol used by prior world model work Burchi & Timofte (2025); Zhang et al. (2025), we focus our main comparison on a 5×10^4 environment-frame budget. For a map that is particularly hard like `3s_vs_5z` it was extend the training to 2×10^5 frames to allow for a clear performance difference. During training, we evaluate every 1,000 steps by computing the median win rate over 30 episodes. The final results are averaged across 3-4 random seeds. For instance, we use observation-dropout augmentation with p=0.03. The action-conditioned CPC horizon is K_{cpc}=8 on all maps except `so_many_baneling` and `2s3z`, where K_{cpc}=5. Baselines include MAPPO Yu et al. (2022), MAMBA Egorov & Shpilman (2022) and MARIE Zhang et al. (2025), MATWM Deihim et al. (2025), and an MBVD variant Xu et al. (2022). We report baseline numbers in Table 2 from their respective papers under the same 5×10^4-frame protocol; we include learning curves only for MACT (Appendix A) because per-seed curves for all baselines are not consistently available in published form.

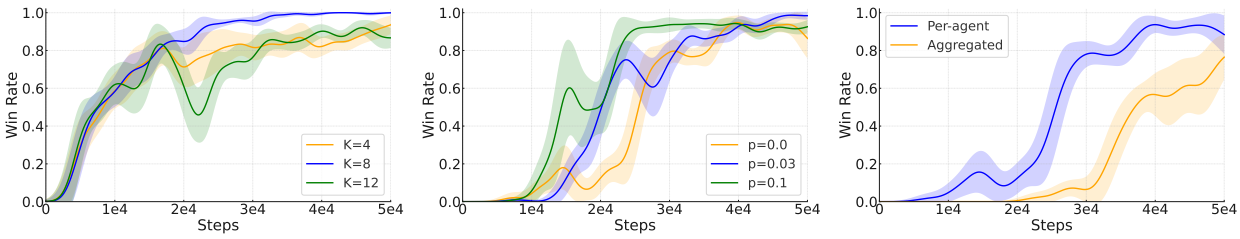
On the evaluated set of maps, MACT attains the highest average performance (Mean row of Table 2: 73.3% vs. 68.0% for MATWM and 63.1% for MARIE) and ranks first on five of the ten maps. Gains are largest on coordination-heavy settings: on `8m` MACT reaches 95.0% median win rate, on `so_many_baneling` 94.4%, and on `MMM` 39.2%. Relative to MARIE, the improvements are +23.0 points on `8m` and +21.4 on `so_many_baneling`. Relative to MATWM, the improvement on `MMM` is +32.2. MACT also converges rapidly on easier maps, achieving 98.7% on `2s_vs_1sc` and 96.7% on `3m`. Differences are small where performance saturates, as in `2m_vs_1z` (95.0% vs. 98.0% for MATWM). On the 200k-frame `3s_vs_5z` evaluation, MACT is comparable to MARIE (65.0% vs. 66.0%).

The learning curves indicate faster rise within the 50k-frame budget on the easier maps and larger seed variance on `MMM`, which is expected given the heterogeneous-unit interactions. MACT trails MATWM on `2s3z`, `3s_vs_3z`, and `3s_vs_4z` (by 15.0, 7.0, and 7.9 points, respectively). These scenarios emphasize short tactical bursts and precise per-agent micro. A single Perceiver aggregation step may propagate less fine-grained context than required, and the shorter K_{cpc}=5 for `2s3z` may contribute. The results support our central claim on the evaluated SMAC regime: coupling a Perceiver's global context with action-conditioned, multi-step prediction improves coordination under tight data budgets. We emphasize that our claims are scoped to the low-data SMAC protocol and the set of maps evaluated here. Broader conclusions about other cooperative benchmarks or higher-budget regimes require additional experiments. Detailed analyses are presented in the following ablation studies.

Comparison to MATWM. MATWM is the most competitive baseline and outperforms MACT on `2s3z`, `3s_vs_3z`, and `3s_vs_4z`, which emphasize short tactical bursts and precise per-agent micro. We hypothesize that MATWM's explicit teammate predictor provides stronger short-horizon cross-agent cues in these scenarios, whereas MACT invests capacity into learning multi-step action-conditioned abstractions that pay off more in coordination-heavy settings (e.g., `8m`, `MMM`, `so_many_baneling`). This suggests a complementary direction: combining long-horizon contrastive supervision with richer cross-agent modeling may further improve performance on micro-intensive maps.

4.1 Ablation Studies

We conduct ablation studies to isolate the impact of MACT's key components. We chose to perform these studies on the `3m` and `8m` maps, as these are representative scenarios where our model achieved great performance. All ablations keep the architecture, optimizer, and implementation protocol fixed to Table 3. For all the conducted experiments, only the factor under test is varied. Panels (a)–(c) in Figure 4 correspond to the three experiments described in this section.



(a) Different K levels. (b) Different noise levels. (c) Per-agent and team aggregated.

Figure 4: Ablations study conducted on 3m and 8m SMAC environments.

CPC horizon. We vary the action-conditioned CPC horizon $K_{\text{cpc}} \in \{4, 8, 12\}$ on `3m` (Figure 4a). The K=8 model rises quickly and reaches the highest plateau, indicating that an 8-step action window is sufficient to cover the reaction–recovery cycle needed for coordinated focus fire. K=4 learns fastest at the very beginning but plateaus lower, consistent with under-specifying longer exchanges. K=12 eventually approaches the K=8 ceiling but starts noticeably slower and exhibits wider variability, reflecting heavier predictors and longer input action windows per InfoNCE term. Overall, an intermediate horizon balances temporal coverage and optimization stability.

Observation dropout. We tested how adding a small amount of noise, by feature-dropout, to the observations affects learning (Figure 4b). We tried feature-dropout probability $p \in \{0.0, 0.03, 0.1\}$ applied only when forming CPC positives on `8m`. Without augmentation (p=0), learning onsets later and saturates lower, suggesting the contrastive task can be solved by brittle token matching. A small corruption (p=0.03) improves both speed and final win rate, pushing the predictor toward action-relevant invariance tied to Perceiver context rather than exact feature memorization. A heavier masking (p=0.1) can accelerate mid-training on some seeds but introduces oscillations and a slight late-stage drop, indicating that excessive

noise erodes import information. A small, targeted amount of augmentation provides the best trade-off, encouraging the model to learn more robust features.

Per-agent vs. team aggregated conditioning. For completeness we test an agent-agnostic variant that replaces own actions with an aggregated joint-action summary at each step:

$$\tilde{a}^{\,i}_{t:t+k-1} \;=\; \mathrm{Agg}\big(a^{1:N}_{t:t+k-1}\big).$$

To keep inputs comparable, the query features are also aggregated to team level ($h_t^{\text{team}} = \frac{1}{N}\sum_i h_t^i$, $e_t^{\text{team}} = \mathrm{Agg}(e_t^{1:N})$) before applying the same Equations equation 5–equation 6.

On `8m`, conditioning the CPC head on each agent's own future actions clearly outperforms using an aggregated joint-action summary (Figure 4c). The per-agent curve lifts off roughly 1×10^4 steps earlier, climbs steadily through mid-training, and reaches ≈ 0.9 median win rate by 5×10^4 steps with tighter seed bands. The aggregated variant lags by about $1-1.5\times10^4$ steps and plateaus lower ($\approx 0.75-0.8$) with higher variance. We attribute this gap to identity-agnostic pooling blurring who executed which maneuver: AC-CPC learns best from a precise mapping between an agent's action window and the evolution of its Perceiver latent, while global team context is already provided by e_t. Adding a coarse joint-action summary appears redundant and noisier, increasing predictor input dimensionality and weakening contrastive alignment. We therefore adopt per-agent conditioning as the default conditioning methodology in MACT.

5 Limitations

MACT is evaluated in the low-data SMAC regime, which we choose because it cleanly isolates the sample-efficiency and coordination challenges that motivate action-conditioned multi-step prediction. Our primary claims are therefore scoped to cooperative micro-management with structured observations and discrete actions under tight interaction budgets. Extending the same training objective and architecture to settings with richer observations (e.g., visual inputs) or continuous control (e.g., Google Research Football with visual observations, or MA-MuJoCo with continuous actions) is a natural next step; doing so mainly requires an appropriate observation encoder/tokenizer and a continuous-action representation for the conditioning window. In addition, ablations are reported on `3m` and `8m` as representative maps where the method is stable and the effects are clearly measurable. Broader ablation coverage may reveal map-dependent sensitivities. We hope future work will test MACT across a wider range of cooperative benchmarks and observation/action modalities to further characterize its generality.

6 Conclusion

MACT is a decentralized action-conditioned transformer world model for MARL. It combines a shared per-agent transformer with a single Perceiver step and an AC-CPC objective to align planned actions with multi-step latent dynamics. On SMAC, MACT outperforms strong model-free and world-model baselines, with the largest gains on coordination-heavy maps. Ablations show that performance is driven by intermediate CPC horizons, light observation dropout, and per-agent action conditioning. MACT integrates naturally into imagination-based MARL pipelines: adding a CPC head and conditioning on short action windows encourages richer long-horizon structure and improves team coordination.

References

Maxime Burchi and Radu Timofte. Learning transformer-based world models with contrastive predictive coding. In *International Conference on Learning Representations (ICLR)*, 2025. URL `https://openreview.net/forum?id=YK9G4Htdew`.

Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). In *International Conference on Learning Representations (ICLR)*, 2016. doi: 10.48550/arXiv.1511.07289. URL `https://arxiv.org/abs/1511.07289`.

Azad Deihim, Eduardo Alonso, and Dimitra Apostolopoulou. Transformer world model for sample efficient multi-agent reinforcement learning, 2025. URL https://arxiv.org/abs/2506.18537.

Vladimir Egorov and Aleksei Shpilman. Scalable multi-agent model-based reinforcement learning. In Piotr Faliszewski, Viviana Mascardi, Catherine Pelachaud, and Matthew E. Taylor (eds.), *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pp. 381–390, Online, 2022. International Foundation for Autonomous Agents and Multiagent Systems (IFAAMAS). URL https://www.ifaamas.org/Proceedings/aamas2022/pdfs/p381.pdf.

Sven Gronauer and Klaus Diepold. Multi-agent deep reinforcement learning: A survey. *Artificial Intelligence Review*, 55(2):895–943, 2022. doi: 10.1007/s10462-021-09996-w. URL https://link.springer.com/article/10.1007/s10462-021-09996-w.

David Ha and Jürgen Schmidhuber. World models, 2018. URL https://arxiv.org/abs/1803.10122.

Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. In *International Conference on Learning Representations (ICLR)*, 2020. URL https://openreview.net/forum?id=S1lOTC4tDS.

Danijar Hafner, Jurgis Pasukonis, Jimmy Ba, and Timothy Lillicrap. Mastering diverse control tasks through world models. *Nature*, 640:647–653, 2025. doi: 10.1038/s41586-025-08744-2. URL https://www.nature.com/articles/s41586-025-08744-2.

Lukasz Kaiser, Mohammad Babaeizadeh, Piotr Milos, Blazej Osinski, Roy H. Campbell, Konrad Czechowski, Dumitru Erhan, Chelsea Finn, Piotr Kozakowski, Sergey Levine, Afroz Mohiuddin, Ryan Sepassi, George Tucker, and Henryk Michalewski. Model-based reinforcement learning for atari, 2019. URL https://arxiv.org/abs/1903.00374.

Jakub Grudzien Kuba, Ruiqing Chen, Muning Wen, Ying Wen, Fanglei Sun, Jun Wang, and Yaodong Yang. Trust region policy optimisation in multi-agent reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2022. URL https://openreview.net/forum?id=EcGGFkNTxdJ.

Michael Laskin, Aravind Srinivas, and Pieter Abbeel. Curl: Contrastive unsupervised representations for reinforcement learning. In *Proceedings of the 37th International Conference on Machine Learning (ICML)*, volume 119 of *Proceedings of Machine Learning Research*, pp. 5639–5650. PMLR, 2020. URL https://proceedings.mlr.press/v119/laskin20a.html.

Dingbang Liu, Fenghui Ren, Jun Yan, Guoxin Su, Wen Gu, and Shohei Kato. Scaling up multi-agent reinforcement learning: An extensive survey on scalability issues. *IEEE Access*, 12:94610–94631, 2024. doi: 10.1109/ACCESS.2024.3410318. URL https://doi.org/10.1109/ACCESS.2024.3410318.

Vincent Micheli, Eloi Alonso, and François Fleuret. Transformers are sample-efficient world models. In *International Conference on Learning Representations (ICLR)*, 2023. URL https://openreview.net/forum?id=vhFu1Acb0xb.

Tabish Rashid, Mikayel Samvelyan, Christian Schroeder de Witt, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. Monotonic value function factorisation for deep multi-agent reinforcement learning. *Journal of Machine Learning Research*, 21(178):1–51, 2020. URL https://jmlr.org/papers/v21/20-081.html.

Jan Robine, Marc Höftmann, Tobias Uelwer, and Stefan Harmeling. Transformer-based world models are happy with 100k interactions. In *International Conference on Learning Representations (ICLR)*, 2023. URL https://openreview.net/forum?id=TdBaDGCpjly.

Mikayel Samvelyan, Tabish Rashid, Christian Schroeder de Witt, Gregory Farquhar, Nantas Nardelli, Tim G. J. Rudner, Chia-Man Hung, Philip H. S. Torr, Jakob Foerster, and Shimon Whiteson. The starcraft multi-agent challenge. In Noa Agmon, Matthew E. Taylor, Edith Elkind, and Manuela Veloso (eds.), *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pp. 2186–2188, Montréal, Canada, 2019. International Foundation for Autonomous Agents and Multiagent

Systems (IFAAMAS). URL https://ifaamas.csc.liv.ac.uk/Proceedings/aamas2019/pdfs/p2186.pdf. Extended Abstract.

Peter Sunehag, Guy Lever, Audrunas Gruslys, Wojciech Marian Czarnecki, Vinicius Zambaldi, Max Jaderberg, Marc Lanctot, Nicolas Sonnerat, Joel Z. Leibo, Karl Tuyls, and Thore Graepel. Value-decomposition networks for cooperative multi-agent learning based on team reward. In Mehrdad Dastani, Gita Sukthankar, Elisabeth André, and Sven Koenig (eds.), *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, pp. 2085–2087, Stockholm, Sweden, 2018. International Foundation for Autonomous Agents and Multiagent Systems (IFAAMAS). URL https://www.ifaamas.org/Proceedings/aamas2018/pdfs/p2085.pdf. Extended Abstract.

Aaron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. Neural discrete representation learning. In *Advances in Neural Information Processing Systems*, volume 30, pp. 6306–6315, 2017. URL https://papers.nips.cc/paper/2017/hash/7a98af17e63a0ac09ce2e96d03992fbc-Abstract.html.

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding, 2018. URL https://arxiv.org/abs/1807.03748.

Jianhao Wang, Zhizhou Ren, Terry Liu, Yang Yu, and Chongjie Zhang. Qplex: Duplex dueling multi-agent q-learning. In *International Conference on Learning Representations (ICLR)*, 2021. URL https://openreview.net/forum?id=Rcmk0xxIQV.

Zhiwei Xu, Dapeng Li, Bin Zhang, Yuan Zhan, Yunpeng Bai, and Guoliang Fan. Mingling foresight with imagination: Model-based cooperative multi-agent reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 35, pp. 11327–11340, 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/49be51578b507f37cd8b5fad379af183-Paper-Conference.pdf.

Chao Yu, Akash Velu, Eugene Vinitsky, Jiaxuan Gao, Yu Wang, Alexandre M. Bayen, and Yi Wu. The surprising effectiveness of ppo in cooperative multi-agent games. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 35, pp. 24611–24624, 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/9c1535a02f0ce079433344e14d910597-Paper-Datasets_and_Benchmarks.pdf.

Weipu Zhang, Gang Wang, Jian Sun, Yetian Yuan, and Gao Huang. Storm: Efficient stochastic transformer based world models for reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 36, pp. 27147–27166, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/5647763d4245b23e6a1cb0a8947b38c9-Abstract-Conference.html.

Yang Zhang, Chenjia Bai, Bin Zhao, Junchi Yan, Xiu Li, and Xuelong Li. Decentralized transformers with centralized aggregation are sample-efficient multi-agent world models. *Transactions on Machine Learning Research*, 2025. URL https://openreview.net/forum?id=4E0lCxBD0U.

Ruijie Zheng, Xiyao Wang, Yanchao Sun, Shuang Ma, Jieyu Zhao, Huazhe Xu, Hal Daumé III, and Furong Huang. Taco: Temporal latent action-driven contrastive loss for visual reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 36, pp. 48203–48225, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/96d00450ed65531ffe2996daed487536-Paper-Conference.pdf.

A Appendix: Training results

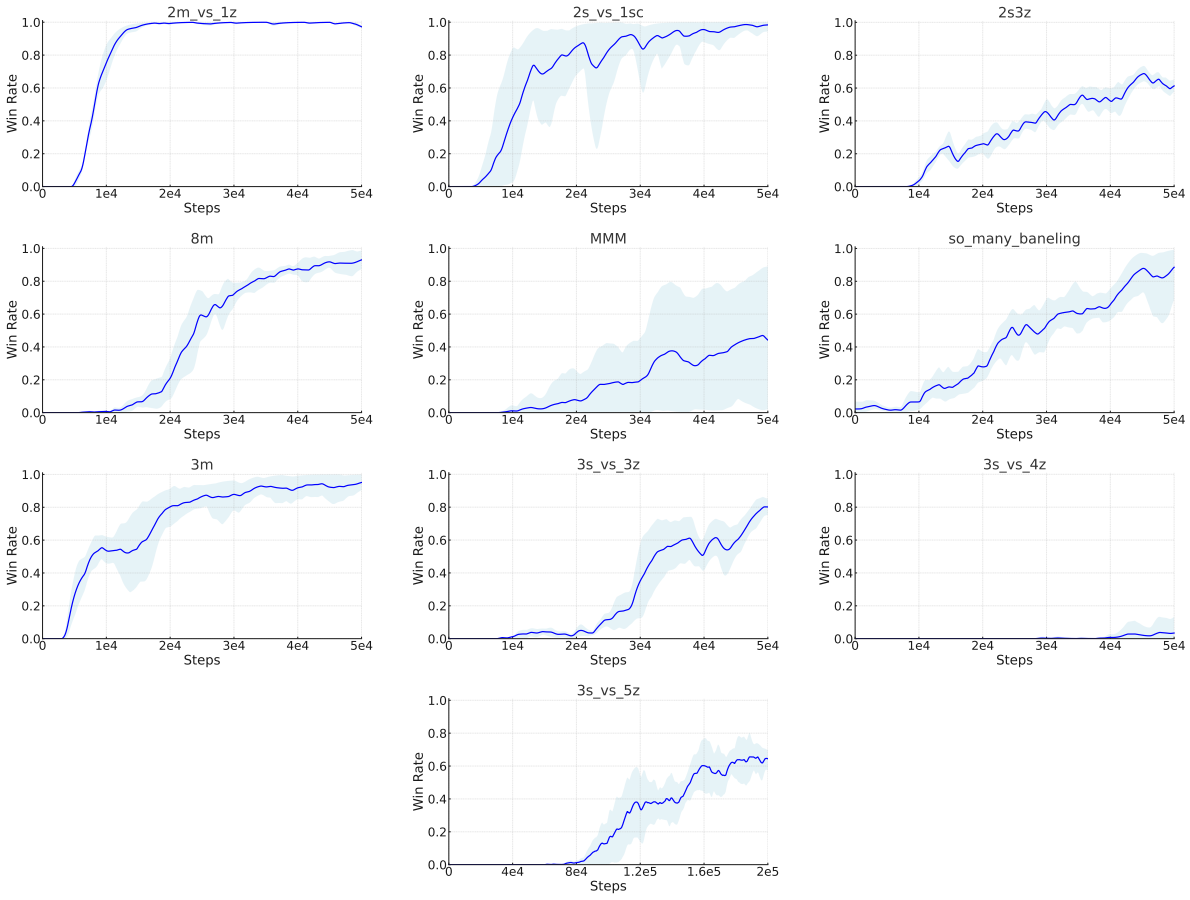


Figure 5: Evaluation win rate across all tested SMAC environments. During training, we ran 30 evaluation episodes every 1,000 environment steps.

B Appendix: MACT implementation

Reproducibility and setup. All ablations are run on the same implementation as our main method and keep MARIE's learning setup wherever applicable. To minimize environment drift, we pin the StarCraft II client to *SC2.4.1.2.60604* (the same version used by MATWM). It is worth to note that the use of different SC2 builds can yield slightly different win rates. Unlike MARIE (Python 3.7), our codebase targets Python 3.11 and PyTorch 2.x. Exact package versions, training scripts, per-map configs, the CSV files used to render plots, and the plotting scripts themselves will be made publicly available upon completion of the review process.

Hyperparameters. We keep global training knobs identical across main and ablation runs (optimizer families, learning rates, clipping, entropy coefficient, etc.; see Table 3). As MARIE, for map-specific settings we select the imagination horizon H and the *number of policy updates* as a function of the number of allied agents N: for $N \leq 3$ (e.g., `2m_vs_1z`, `3m`, `3s_vs_5z`) we use H=15 and 4 updates; for $4 \leq N \leq 5$ (e.g., `2s3z`) we use H=8 and 10 updates; for $N \geq 6$ (e.g., `8m`, `MMM`, `so_many_baneling`) we use H=5 and 30 updates. Unless stated otherwise, the tokenizer and world model are each trained for 200 epochs, we perform 5 PPO epochs per policy update, and the λ-return uses λ=0.95 with γ=0.99. Map-specific overrides (e.g., K_{cpc}) follow Table 3.

Table 3: Hyperparameters for MACT in SMAC environments.

Hyperparameter	**Value**
Batch size for tokenizer training	256
Batch size for world model training	30
Optimizer for tokenizer	AdamW
Optimizer for world model	AdamW
Optimizer for actor & critic	Adam
Tokenizer learning rate	0.0003
World model learning rate	0.0003
Actor learning rate	0.0005
Critic learning rate	0.0005
Gradient clipping (actor & critic)	max-norm 1
Gradient clipping (tokenizer)	10
Gradient clipping (world model)	max-norm 1
Weight decay for world model	0.01
λ for λ-return	0.95
Discount factor γ	0.99
Entropy coefficient	0.001
Buffer size (transitions)	2.5×10^5
Tokenizer training epochs	200
World model training epochs	200
Collected transitions between updates	{100, 200}
Epochs per policy update (PPO epochs)	5
PPO clipping parameter ϵ	0.2
Number of imagined rollouts	600 or 400[‡]
Imagination horizon H	{15, 8, 5}
Number of policy updates	{4, 10, 30}
Number of stacking observations	5
Observe agent id	False
Observe last action of itself	False
AC-CPC	
CPC horizon K_{cpc}	8[†]
Geometric decay λ_{cpc}	0.75
Projection dimension (query/key)	512
Activation	ELU Clevert et al. (2016)
Action-conditioning	Per-agent action window $a^i_{t:t+k-1}$
Negatives (InfoNCE)	In-batch
Positive target	Perceiver latent of augmented obs.
Vector-state augmentation	dropout p=0.03

[†] Particularly, we used K_{cpc}=5 on `so_many_baneling`, `2c_vs_64zg` and `2s3z`; 8 on all other maps.
[‡] We follow MARIE's SMAC configuration, which uses 600 or 400 imagined rollouts depending on the map and agent count.

Pseudocode overview. Algorithm 1 summarizes MACT in three phases. (i) *Experience collection* gathers short trajectories in SMAC and stores $\{o, a, r, \gamma\}$ in a replay buffer. (ii) *World–model update* samples a segment, builds per-step team context with one Perceiver cross-attention to obtain $e_t^{1:N}$, and runs the shared per-agent Transformer to produce h_t^i. The one-step likelihood (token, reward, discount, and optional availability heads) is combined with a *per-agent* action-conditioned CPC loss: for k=0:$K_{\text{cpc}}-1$, a predictor consumes $\left[h_t^i; e_t^i; a^i_{t:t+k-1}\right]$ (for k=0 the action window is empty) and is trained via InfoNCE against a *projected* Perceiver latent from an augmented future view; terms are weighted geometrically by λ_{cpc}. (iii) *Policy learning in imagination (CTDE)* freezes the world model and unrolls H_{roll} latent steps; masked decentralized actors produce actions, a centralized critic evaluates returns, and λ-returns train the agents.

Computational complexity. Let N be the number of agents, H the sampled segment length, K_{cpc} the CPC horizon, Q the number of (time,agent) positives in the minibatch, and d_z the projection width. MACT preserves MARIE's near-linear scaling in team size: the world-model backbone still consists of (i) per-agent

Algorithm 1: MACT training pseudocode (per-agent)

Input : Replay buffer $\mathcal{D}$;
horizons H (model), H_{roll} (imagination);
CPC horizon K_{cpc};
decay λ_{cpc}
Modules: VQ tokenizer encoder E;
shared per-agent Transformer ϕ;
Perceiver cross-attn θ;
heads: obs-token, reward, discount, avail;
CPC heads $\{p_k, q_k\}$;
actors $\{\pi^i_\psi\}$;
critic V_ξ
for *epoch* $= 1, 2, \ldots$ **do**
  $o \leftarrow$ ENV.RESET() `// (i) Collect real experience`
  repeat
    sample $a^i_t \sim \pi^i_\psi(o^i_t)$ (mask infeasible)
    $(o', r, \texttt{done}, m) \leftarrow$ ENV.STEP$(a^{1:N})$ `// m_t is avail mask`
    push $\{o, a, r, \texttt{done}, m\}$ to $\mathcal{D}$;
    $o \leftarrow \texttt{done}?$ ENV.RESET() $: o'$
  until n *steps*
  sample segment $\{o^{1:N}_t, a^{1:N}_t, r_t, c_t, m_t\}^{\tau+H-1}_{t=\tau} \sim \mathcal{D}$ `// (ii) Train world model`
  $x^i_t \leftarrow E(o^i_t)$ for all t, i `// vector → discrete tokens`
  for $t = \tau$ **to** $\tau + H - 1$ **do**
    $e^{1:N}_t \leftarrow \text{PERCEIVER}_\theta(\{x^i_t\}, a^{1:N}_t)$
    $h^i_t \leftarrow \text{TRANSFORMER}_\phi([x^i_{\le t}, a^i_{\le t}, e^i_{\le t}])$
    for $j = 1$ **to** K_{tok} **do**
      predict $\hat{x}^i_{t+1,j}$ autoregressively conditioning on $\hat{x}^i_{t+1,<j}$
    predict $\hat{r}^i_t$, $\hat{c}^i_t$, $\hat{m}^i_{t+1}$ from h^i_t
  build boundary mask $\mathcal{M}$ on flattened streams $y^i_{1:T}$
  $\mathcal{L}_{\text{dyn}} \leftarrow \mathcal{L}_{\text{token}}(\mathcal{M}) + \mathcal{L}_{\text{reward}} + \mathcal{L}_{\text{discount}} + \mathcal{L}_{\text{av}}$
  $o'_t \leftarrow$ AUGMENT_VECTOR(o_t); `// Per-agent AC-CPC (augmented future Perceiver latents)`
  $x'_t \leftarrow E(o'_t)$
  ${e'_t}^{1:N} \leftarrow \text{PERCEIVER}_\theta(\{{x'_t}^i\}, a^{1:N}_t)$ for all t
  for $k = 0$ **to** $K_{cpc}-1$ **do**
    foreach (i, t) *in minibatch* **do**
      $\hat{z}^i_{t+k} \leftarrow p_k([h^i_t; e^i_t; a^i_{t:t+k-1}])$
      $z^i_{t+k} \leftarrow q_k({e'}^i_{t+k})$
    compute InfoNCE ℓ_k with in-batch negatives
  $$\mathcal{L}_{\text{cpc}} \leftarrow \sum_{k=0}^{K_{\text{cpc}}-1} \frac{\lambda^k_{\text{cpc}}}{\sum_{j=0}^{K_{\text{cpc}}-1} \lambda^j_{\text{cpc}}} \ell_k$$
  Update $(E, \phi, \theta, \{p_k, q_k\})$ on $\mathcal{L}_{\text{WM}} = \mathcal{L}_{\text{dyn}} + \mathcal{L}_{\text{cpc}}$
  init $(o_0, a_0) \sim \mathcal{D}$;
  $x^i_0 \leftarrow E(o^i_0)$; `// (iii) Train actors in imagination (CTDE)`
  $e^{1:N}_0 \leftarrow \text{PERCEIVER}_\theta(\{x^i_0\}, a^{1:N}_0)$;
  $h^i_0 \leftarrow \text{TRANSFORMER}_\phi([x^i_{\le 0}, a^i_{\le 0}, e^i_{\le 0}])$
  for $t = 0$ **to** $H_{roll} - 1$ **do**
    sample $a^i_t \sim \pi^i_\psi(\hat{o}^i_t)$ (mask infeasible)
    world model predicts $\hat{r}_t, \hat{c}_t$, and autoregressively predicts $\hat{x}^i_{t+1}$
    $\hat{o}^i_{t+1} \leftarrow D(\hat{x}^i_{t+1})$ `// decode predicted tokens`
    update $(e^{1:N}_{t+1}, h^{1:N}_{t+1})$ from $(\hat{x}^{1:N}_{t+1}, a^{1:N}_{t+1})$
  update ψ by advantage PG and ξ by symlog MSE

local Transformer dynamics (no joint cross-agent self-attention over all tokens) and (ii) a single Perceiver-style cross-attention step over the current-step token set of length $L = N(K+1)$. Relative to MARIE, the only additional cost comes from AC-CPC. For each $k \in \{0, \ldots, K_{\text{cpc}} - 1\}$, MACT applies per-agent MLP projections $p_k(\cdot)$ and $q_k(\cdot)$, giving an added projection cost $O\Big(NH \sum_{k=0}^{K_{\text{cpc}}-1} C_{\text{MLP}}(k)\Big)$, where $C_{\text{MLP}}(k)$ depends on the input width $D_x+D_e+k|\mathcal{A}|$. The InfoNCE term computes a batched similarity between Q queries and Q keys; implemented as a dense matrix multiply this costs $O(Q^2 d_z)$ per k. Overall, AC-CPC

adds an approximately linear-in-NHK_{cpc} projection term plus a quadratic-in-Q similarity term per horizon step, while leaving the backbone architecture unchanged. In practice, Q is the minibatch size of agent–time pairs and is typically much smaller than the sequence lengths that dominate Transformer attention, so the additional overhead is moderate and does not alter the scaling behavior with N.

What to toggle for each ablation. We change one switch at a time and keep everything else fixed. Action–conditioning is either per-agent or team (flag: `cpc_mode`). The CPC horizon and its geometric weighting use `K_cpc` and λ_{cpc} (defaults 8 and 0.75. Map-specific overrides are in Table 3). InfoNCE key detachment is a simple on/off (`detach_keys`). Reward targets use SmoothL1 on symlogged values by default. MARIE-style discrete two-hot is enabled via `use_ce_for_reward` (only used for `2m_vs_1z`). All ablations use the same evaluation protocol and plotting code.