

REDISTRIBUTING REWARDS ACROSS TIME AND AGENTS FOR MULTI-AGENT REINFORCEMENT LEARNING

Aditya Kapoor^{*}, Kale-ab Tessera², Harshad Khadilkar³, Mayank Baranwal³, Jan Peters⁴, Stefano Albrecht², and Mingfei Sun¹

¹University of Manchester

²University of Edinburgh

³IIT, Bombay

⁴TU Darmstadt

ABSTRACT

Credit assignment—disentangling each agent’s contribution to a shared reward—is a critical challenge in cooperative multi-agent reinforcement learning (MARL). To be effective, credit assignment methods must preserve the environment’s optimal policy. Some recent approaches attempt this by enforcing return equivalence, where the sum of distributed rewards must equal the team reward. However, their guarantees are conditional on a learned model’s regression accuracy, making them unreliable in practice. We introduce Temporal-Agent Reward Redistribution (TAR²), an approach that decouples credit modeling from this constraint. A neural network learns unnormalized contribution scores, while a separate, deterministic normalization step enforces return equivalence by construction. We demonstrate that this method is equivalent to a valid Potential-Based Reward Shaping (PBRS), which guarantees the optimal policy is preserved regardless of model accuracy. Empirically, on challenging SMACLite and Google Research Football (GRF) benchmarks, TAR² accelerates learning and achieves higher final performance than strong baselines. These results establish our method as an effective solution for the agent-temporal credit assignment problem. [Github Code](#)

1 INTRODUCTION

MARL (Albrecht et al., 2024) is a powerful paradigm for solving complex cooperative tasks, with landmark successes in domains ranging from logistics and robotics to challenging games (Krnjaic et al., 2023; Sartoretti et al., 2019; Vinyals et al., 2019; Kurach et al., 2020). In these settings, teams of agents must learn to synchronize their actions to achieve a shared goal.

Despite this progress, a key bottleneck is the multi-agent credit assignment problem: allocating a shared team reward to guide agent learning. The challenge is amplified in episodic MARL where a single feedback signal arrives only at a trajectory’s end. Here, agents struggle to resolve two coupled problems: when their critical contributions occurred (temporal credit assignment) and which agents were responsible (agent credit assignment).

For credit assignment methods to be effective, they should not alter the environment’s optimal policy. Some recent approaches, like STAS and AREL, attempt this by enforcing a return equivalence constraint, where distributed rewards must sum to the team reward. However, this approach is theoretically fragile. They train a single model to both predict credit and satisfy this constraint simultaneously. Consequently, their policy-invariance guarantee is conditional on the network being a perfect regressor—a condition practically unattainable. Any prediction error breaks the guarantee and risks leading to suboptimal policies.

^{*}Email: aditya.kapoor@postgrad.manchester.ac.uk

To solve this fragility, we introduce Temporal-Agent Reward Redistribution (TAR²), a method designed for structural robustness, which we illustrate in Figure 1. The core idea is to **decouple credit modeling from constraint satisfaction**. A neural network performs a single, focused task: learning unnormalized scores representing the *absolute* importance of each agent’s actions. A separate, deterministic normalization step then makes this importance *relative*, converting the scores into a valid probability distribution used to construct the final rewards. This guarantees return equivalence by construction, irrespective of the network’s accuracy. We demonstrate that this two-step architecture is equivalent to a valid Potential-Based Reward Shaping (PBRs) (Ng, 1999), which provides the formal guarantee that the optimal policy is preserved.

2 BACKGROUND

We formalize the cooperative MARL problem as a Decentralized Partially Observable Markov Decision Process (Dec-POMDP) (Oliehoek & Amato, 2016) and review Potential-Based Reward Shaping (PBRs) (Ng, 1999), the theoretical foundation for our approach.

2.1 PROBLEM FORMULATION

A fully cooperative multi-agent task is a Dec-POMDP (Oliehoek & Amato, 2016; Amato, 2024), defined by the tuple $\mathcal{M} = \langle \mathcal{S}, \{\mathcal{A}_i\}_{i=1}^N, \mathcal{P}, \{\Omega_i\}_{i=1}^N, \mathcal{O}, \mathcal{R}, \rho_0, \gamma, N \rangle$. The environment consists of N agents interacting in a global state space \mathcal{S} with an initial state distribution ρ_0 . At each timestep t , the team takes a joint action $\mathbf{a}_t = \{a_{1,t}, \dots, a_{N,t}\}$ from the joint action space $\mathcal{A} = \times_{i=1}^N \mathcal{A}_i$, which causes a transition to a new state s_{t+1} according to $\mathcal{P}(s_{t+1}|s_t, \mathbf{a}_t)$. Due to partial observability, agent i receives only a local observation $o_{i,t} \in \Omega_i$ from the observation function \mathcal{O} .

Because a single observation is often insufficient to disambiguate the true state, each agent must condition its policy on its local action-observation history, $\tau_{i,t} = (o_{i,0}, a_{i,0}, \dots, o_{i,t-1}, a_{i,t-1}, o_{i,t})$ (Amato, 2024). For notational convenience, we denote the histories of all other agents as $\tau_{-i,t} = \{\tau_{j,t}\}_{j \neq i}$, and the joint history of all agents as $\tau_t = (\tau_{1,t}, \dots, \tau_{N,t})$. In a decentralized execution setting, each agent learns a policy $\pi_i(a_{i,t}|\tau_{i,t})$ that depends only on its own history. The agents’ policies combine to form a joint policy $\pi = \prod_{i=1}^N \pi_i$. The team receives a shared reward $r_t = \mathcal{R}(s_t, \mathbf{a}_t)$ and, with a discount factor γ , aims to learn a joint policy π that maximizes the expected discounted return: $J(\pi) = \mathbb{E}_{\tau \sim \pi, \rho_0} \left[\sum_{t=0}^T \gamma^t r_t \right]$, where τ denotes a full episode trajectory. To learn effective decentralized policies in large state-action spaces, we adopt the Centralized Training with Decentralized Execution (CTDE) paradigm (Foerster et al., 2018; Lowe et al., 2017). During centralized training, agents leverage global information to guide learning, while at test-time they execute policies using only local information.

We focus on the challenging episodic setting, where reward is only dispensed at the end of an episode $r_t = 0$ for $t < T$, and $r_t = R(s_T)$ is a terminal reward. This sparsity makes credit assignment exceptionally difficult, as agents must deduce which actions in a long trajectory led to the final outcome from an episodic signal. While policies are executed decentrally, we assume a centralized training paradigm where a shaping function can access the joint history τ_t to guide learning.

2.2 POTENTIAL-BASED REWARD SHAPING

To create dense rewards without distorting the underlying problem, we leverage PBRs (Ng, 1999). In single-agent RL, PBRs augments the environment’s reward $\mathcal{R}(s, a)$ by an additional shaping reward, $F(s, a, s') = \gamma \Phi(s') - \Phi(s)$, using a potential function $\Phi : \mathcal{S} \rightarrow \mathbb{R}$ to improve the speed of convergence. The key property of PBRs is that it guarantees *policy invariance* – any optimal policy in the shaped-reward MDP remains optimal in the original MDP. This guarantee extends to the fully cooperative multi-agent setting, preserving the set of optimal joint policies (Devlin et al., 2014) and forming a theoretically sound basis for credit assignment.

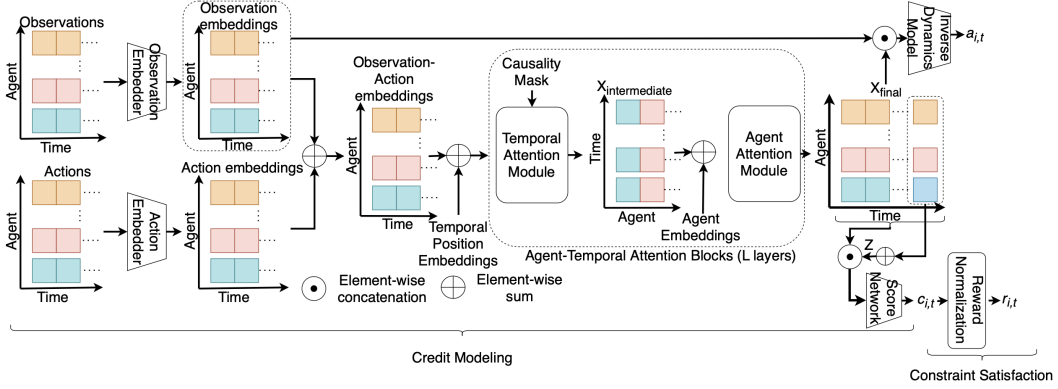


Figure 1: The TAR² architecture processes trajectory data through four main stages. (1) Input sequences are converted into embeddings with positional encoding. (2) A multi-layer transformer block with sequential Temporal and Agent Attention builds context-aware representations, regularized by an auxiliary Inverse Dynamics task to ensure causality. (3) The Score Network computes unnormalized scores by conditioning each timestep’s representation on a learned Final Outcome Embedding (Z). (4) A final Probabilistic Normalization step converts these scores and the global reward $R(s_T)$ into dense, per-agent rewards $\{r_{i,t}\}$ that satisfy strict return equivalence.

3 TEMPORAL-AGENT REWARD REDISTRIBUTION

As established, PBRS provides a sound theoretical basis for policy-preserving credit assignment. However, its practical application is not straightforward. The guarantee of policy invariance holds for *any* potential function, but an unconstrained or poorly learned one can introduce significant noise and high variance into the policy gradients, destabilizing and slowing down convergence.

Our method, Temporal-Agent Reward Redistribution (TAR²), is designed to harness the guarantees of PBRS while mitigating these practical instabilities. The core idea is to decouple credit modeling from constraint satisfaction. As illustrated in Figure 1, a neural network performs a single, focused task: learning unnormalized contribution scores. A separate, deterministic normalization step then constructs the final rewards, guaranteeing return equivalence by construction. In this section, we formalize this design, prove its theoretical guarantees, analyze its learning dynamics, and detail the architecture that operationalizes these principles.

3.1 REWARD REDISTRIBUTION FORMULATION

We formalize our two-step process as follows. The reward model, parameterized by θ , learns unnormalized contribution scores, $c_{i,t}$. We then use a deterministic shift-and-normalize scheme to convert these scores into weights. The weights convert the scores to final shaped rewards, $r_{i,t}$, enforcing strict return equivalence ($\sum_{t,i} r_{i,t} = R(s_T)$) by construction. This structural guarantee is critical for reducing the variance in the learning signal (Sec 3.4.1), distinguishing TAR² from prior work where policy invariance is a fragile and conditional property. First, we compute temporal weights,

$$w_t^{\text{temp}} = \frac{c_t^{\text{agg}} - \min_{t'} c_{t'}^{\text{agg}}}{\sum_{t''} (c_{t''}^{\text{agg}} - \min_{t'} c_{t'}^{\text{agg}}) + \epsilon}, \quad (1)$$

by normalizing aggregated scores ($c_t^{\text{agg}} = \sum_i c_{i,t}$) across the trajectory. We then compute agent-specific weights,

$$w_{i,t}^{\text{agent}} = \frac{c_{i,t} - \min_j c_{j,t}}{\sum_k (c_{k,t} - \min_j c_{j,t}) + \epsilon}, \quad (2)$$

by normalizing the individual scores within each timestep. The indices iterate over active agents and timesteps, and ϵ (e.g. 1e-8) is a small constant for numerical stability. The final redistributed reward is then constructed as

$$s_{i,t} = w_t^{\text{temp}} w_{i,t}^{\text{agent}} R(s_T). \quad (3)$$

3.2 OPTIMAL POLICY PRESERVATION

We prove TAR² preserves the optimal policy by framing it within multi-agent PBRS. Let $r_{i,t}^{\text{orig}}$ be agent i 's ground-truth contribution to the team reward r_t^{orig} . Our model produces a dense, shaped reward $s_{i,t} = w_t^{\text{temp}} w_{i,t}^{\text{agent}} R(s_T)$. While the complete potential-based reward is $r'_{i,t} = s_{i,t} + r_{i,t}^{\text{orig}}$, TAR² uses only the shaped reward $s_{i,t}$ for learning. This is a critical design choice: because $r_{i,t}^{\text{orig}}$ is zero for all $t < T$ in the episodic setting, dropping it eliminates a sparse, high-variance signal. As we mathematically justify in Sec 3.4.1, this significantly reduces variance and stabilizes learning.

Proposition 3.1 (Optimal Policy Preservation). *Let \mathcal{M}_{env} be a Dec-POMDP where agent i receives reward $r_{i,t}^{\text{orig}}$. Let $\mathcal{M}_{\text{TAR}^2}$ be an identical environment where agent i receives the augmented reward $r'_{i,t} = r_{i,t}^{\text{orig}} + s_{i,t}$. Any joint policy π^* optimal in $\mathcal{M}_{\text{TAR}^2}$ is also optimal in \mathcal{M}_{env} .*

Proof. The proof demonstrates that $s_{i,t}$ is a valid per-agent potential-based shaping reward. As established by [Devlin & Kudenko \(2011\)](#), using such individual potential functions preserves the set of optimal joint policies. For each agent i , we define a history-based potential function $\Phi_i(\tau_t) = \sum_{k=0}^{t-1} s_{i,k}$ (assuming $\gamma = 1$ for the episodic setting). The corresponding shaping function is:

$$F_{i,t} = \gamma \Phi_i(\tau_{t+1}) - \Phi_i(\tau_t) = \sum_{k=0}^t s_{i,k} - \sum_{k=0}^{t-1} s_{i,k} = s_{i,t} \quad (4)$$

Since $s_{i,t}$ adheres to the PBRS condition, the transformation is policy-invariant. The expected total return in the shaped environment is:

$$J_{\text{PBRS}}(\pi) = \mathbb{E}_{\pi} \left[\sum_{t,i} (r_{i,t}^{\text{orig}} + s_{i,t}) \right] = J_{\text{env}}(\pi) + \mathbb{E}_{\pi} \left[\sum_{t,i} s_{i,t} \right] \quad (5)$$

By construction, the total shaping reward $\sum_{t,i} s_{i,t} = R(s_T)$. Since $J_{\text{env}}(\pi) = \mathbb{E}_{\pi}[R(s_T)]$, the new objective is $J_{\text{PBRS}}(\pi) = 2J_{\text{env}}(\pi)$. As this is a constant scaling of the original objective, the set of optimal policies is preserved. \square

3.3 ANALYSIS OF GRADIENT DYNAMICS

Beyond guaranteeing optimality, we now analyze how TAR² influences the learning dynamics. We prove that while TAR² introduces a beneficial bias to the joint policy gradient, it preserves the gradient direction for each individual agent.

Proposition 3.2 (Stochastic Gradient Direction Preservation). *For any agent k and any sampled trajectory τ , the stochastic policy gradient estimate under TAR²'s rewards,*

$$\hat{\mathbf{g}}_{k,\text{TAR}^2}(\tau) = \delta_k(\tau) \hat{\mathbf{g}}_{k,\text{Global}}(\tau), \quad (6)$$

is proportional to the gradient estimate under the original team reward, $\hat{\mathbf{g}}_{k,\text{Global}}(\tau)$. The scaling factor $\delta_k(\tau) \in [0, 1]$ is a trajectory-dependent scalar.

Proof. The policy gradient estimate for agent k is the product of its score function and the total return it receives. Under the global episodic reward, $R(s_T)$, the estimate is

$$\hat{\mathbf{g}}_{k,\text{Global}}(\tau) = G_k(\tau) R(s_T), \quad (7)$$

with the score function $G_k(\tau) = \sum_{t=0}^{T-1} \nabla_{\theta_k} \log \pi_{\theta_k}(a_{k,t} | \tau_{k,t})$. Under TAR², the estimate uses the agent's individual return, $R_k(\tau) = \sum_{t=0}^{T-1} r_{k,t}$, resulting in

$$\hat{\mathbf{g}}_{k,\text{TAR}^2}(\tau) = G_k(\tau) R_k(\tau). \quad (8)$$

By substituting our definition of $r_{k,t}$ from Eq. 3, we find that $R_k(\tau)$ is a scaled version of the global reward

$$R_k(\tau) = \left(\sum_{t=0}^{T-1} w_{k,t}^{\text{agent}} w_t^{\text{temp}} \right) R(s_T). \quad (9)$$

Letting $\delta_k(\tau) = \sum_t w_{k,t}^{\text{agent}} w_t^{\text{temp}}$, the relationship becomes $\hat{\mathbf{g}}_{k,\text{TAR}^2}(\tau) = \delta_k(\tau) \hat{\mathbf{g}}_{k,\text{Global}}(\tau)$. Since weights $w_{k,t}^{\text{agent}} \in [0, 1]$ and $w_t^{\text{temp}} \in [0, 1]$ with $\sum_t w_t^{\text{temp}} = 1$, the scalar $\delta_k(\tau) = \sum_t w_t^{\text{temp}} w_{k,t}^{\text{agent}} \leq \sum_t w_t^{\text{temp}} = 1$. As all weights are non-negative, $\delta_k(\tau) \geq 0$. Thus, $\delta_k(\tau) \in [0, 1]$, and the stochastic gradient direction for agent k is preserved. \square

Implications for Joint Policy Convergence. Crucially, while TAR² preserves the gradient direction for *each agent individually*, the joint policy gradient, $\mathbf{G}_{\text{TAR}^2} = \sum_k \delta_k(\tau) \hat{\mathbf{g}}_k$, is not parallel to the true joint gradient, $\mathbf{G}_{\text{Global}} = \sum_k \hat{\mathbf{g}}_k$. This deviation introduces a *beneficial bias* – TAR² trades the unbiased but high-variance true gradient for a lower-variance, biased estimate that credits agents proportionally to their learned contribution. While this informed bias may lead the parameters to a different convergent point $\theta_{\text{TAR}^2}^*$ (Devlin et al., 2014), our PBRs guarantee ensures the resulting policy, $\pi(\cdot; \theta_{\text{TAR}^2}^*)$, remains in the set of optimal policies. This provides a structural robustness that methods reliant on unconstrained regression targets lack.

3.4 VARIANCE REDUCTION PROPERTIES OF TAR²

While our framework guarantees that the optimal policy is preserved (Sec 3.2), this alone does not guarantee efficient learning. The standard PBRs formulation allows for any potential function, which can introduce significant noise and high variance into the policy gradients, leading to slow and unstable convergence. TAR² is explicitly designed to mitigate this issue through two primary mechanisms, which we analyze below.

3.5 VARIANCE REDUCTION FROM STRUCTURAL CONSTRAINTS

The core design choice of TAR² is to use only the shaped reward $s_{i,t}$ for learning, rather than the full potential-based reward $r'_{i,t} = r_{i,t}^{\text{orig}} + s_{i,t}$. This choice structurally reduces the variance of the joint policy gradient estimator.

Let the score function for agent k be $G_k(\tau) = \sum_{t=0}^{T-1} \nabla_{\theta_k} \log \pi_{\theta_k}(a_{k,t} | \tau_{k,t})$. The joint policy gradient estimator under a full PBRs formulation would be $\hat{g}_{\text{PBRs}}(\tau) = \sum_k G_k(\tau)(R_k^{\text{orig}}(\tau) + S_k(\tau))$, where R_k^{orig} and S_k are the returns for agent k from the original and shaped rewards, respectively. This estimator can be decomposed into two parts: a gradient component from the original rewards, $\hat{g}_{\text{orig}}(\tau) = \sum_k G_k(\tau)R_k^{\text{orig}}(\tau)$, and the TAR² estimator from our shaped rewards, $\hat{g}_{\text{TAR}^2}(\tau) = \sum_k G_k(\tau)S_k(\tau)$.

Using the decomposition for the variance of a sum, the variance of the full PBRs estimator is $\text{Var}(\hat{g}_{\text{PBRs}}) = \text{Var}(\hat{g}_{\text{orig}}) + \text{Var}(\hat{g}_{\text{TAR}^2}) + 2\text{Cov}(\hat{g}_{\text{orig}}, \hat{g}_{\text{TAR}^2})$. The term $\text{Var}(\hat{g}_{\text{orig}})$ is the primary source of instability. In the episodic setting, the per-agent return $R_k^{\text{orig}}(\tau)$ is a sparse, high-variance signal, making \hat{g}_{orig} a noisy estimator for the joint policy gradient. Since $\text{Var}(\hat{g}_{\text{orig}})$ is a non-negative term, it follows that $\text{Var}(\hat{g}_{\text{PBRs}}) > \text{Var}(\hat{g}_{\text{TAR}^2})$.

By using only the shaped rewards $s_{i,t}$ for learning, TAR² structurally eliminates the noisy estimator component \hat{g}_{orig} . This provides a denser, lower-variance signal for more stable and efficient learning, while the policy-invariance guarantee is retained because the total redistributed reward equals the original team reward ($\sum_{t,i} s_{i,t} = R(s_T)$).

3.5.1 VARIANCE REDUCTION FROM FINAL-STATE CONDITIONING

Second, our use of final-state conditioning provides a more causally-correct and lower-variance learning target. Our reward model predicts contribution scores, $c_{i,t}$, conditioned on the final outcome of the trajectory, Z (e.g., the terminal state). By the Law of Total Variance, the variance of these scores can be decomposed as

$$\underbrace{\text{Var}(c_{i,t} | \tau_t)}_{\text{Original Var.}} = \underbrace{\mathbb{E}[\text{Var}(c_{i,t} | \tau_t, Z)]}_{\text{Remaining Var.}} + \underbrace{\text{Var}(\mathbb{E}[c_{i,t} | \tau_t, Z])}_{\text{TAR}^2 \text{ Target Var.}} \quad (10)$$

Our model’s learning target is the final term, the expected contribution $\mathbb{E}[c_{i,t} | \tau_t, Z]$. Since variance is non-negative, this proves that the variance of our learning target is less than or equal to the variance of the unconditioned signal

$$\text{Var}(\mathbb{E}[c_{i,t} | \tau_t, Z]) \leq \text{Var}(c_{i,t} | \tau_t). \quad (11)$$

By learning this less noisy, post-hoc signal, TAR² benefits from a sharper and more stable learning target.

The introduction of the final outcome, Z , means our potential function is implicitly conditioned on information from the end of the trajectory, i.e., $\Phi(\tau_t, Z)$. This does not invalidate the PBRs

guarantees. The policy invariance proof relies on the telescoping sum of potential differences, which holds even when the potential is conditioned on a variable, Z , that is constant for any given trajectory (Devlin et al., 2014; Arjona-Medina et al., 2019). Since this is a valid choice for the potential function’s state representation, all policy preservation guarantees remain intact.

3.6 MODEL ARCHITECTURE AND TRAINING

We now detail the architecture that operationalizes our TAR² method. As depicted in Figure 1, our model is a sequence-to-sequence network that processes the joint action-observation history, τ , to produce the unnormalized contribution scores, $c_{i,t}$.

3.6.1 ARCHITECTURE DETAILS

Our reward model uses a dual-transformer design to capture both temporal and inter-agent dependencies. As shown in Figure 1, the architecture integrates three key components:

Final-State Conditioning. To ground credit assignment in the trajectory’s outcome, the entire model is conditioned on an embedding of the final state, Z . As justified in our variance analysis (Sec 3.5.1), this provides a provably lower-variance learning target.

Inverse Dynamics for Causal Representations. To ensure the embeddings are causally relevant, an auxiliary inverse dynamics model regularizes the shared representations and improve downstream task performance (Pathak et al., 2017; Brandfonbrener et al., 2023). A separate MLP prediction head takes the concatenated latent embeddings from two consecutive timesteps, $\text{embed}(s_{i,t})$ and $\text{embed}(s_{i,t+1})$, and outputs a probability distribution, π_{ID} , over the action space. It is trained via an auxiliary loss to predict the agent’s action, $a_{i,t}$. This forces the shared embeddings to encode controllable aspects of the state, preventing the credit assignment from relying on spurious correlations.

Deterministic Reward Normalization. Finally, the raw scores $c_{i,t}$ from the transformer body are passed to the deterministic normalization function (Sec 3.1). This non-learned step is the structural property that guarantees the strict return equivalence required for effectively learning and preserving the environment’s optimal policy.

3.6.2 TRAINING OBJECTIVE

The reward model, parameterized by θ , is trained by minimizing a composite loss function. The objective provides a strong learning signal for identifying the *importance* of each agent’s actions by regressing the sum of scores against the true episodic reward, while an inverse dynamics term regularizes the representations. The resulting objective

$$\mathcal{L}(\theta) = \mathbb{E}_{\tau \sim \mathcal{B}} \left[\underbrace{\left(R(s_T) - \sum_{t,i} c_{i,t} \right)^2}_{\text{Reward Regression Loss}} - \lambda \underbrace{\sum_{t,i} \log \pi_{\text{ID}}(a_{i,t})}_{\text{Inverse Dynamics Regularizer}} \right] \quad (12)$$

combines this regression loss with the inverse dynamics regularizer, modulated by a hyperparameter λ . This objective trains the model to assign a higher total score to better trajectories. The separate normalization step (Sec 3.1) then takes these meaningfully-scaled scores and distributes their value as credit in a way that structurally guarantees policy invariance.

4 RELATED WORK

Our work addresses the joint agent-temporal credit assignment problem. We position TAR² within the existing literature, highlighting how our design overcomes the key limitations of prior work, particularly their reliance on theoretically brittle learning schemes.

4.1 TEMPORAL CREDIT ASSIGNMENT

Temporal credit assignment aims to transform a single, sparse episodic reward into a sequence of per-timestep signals. Prominent single-agent approaches include analyzing state-value differences

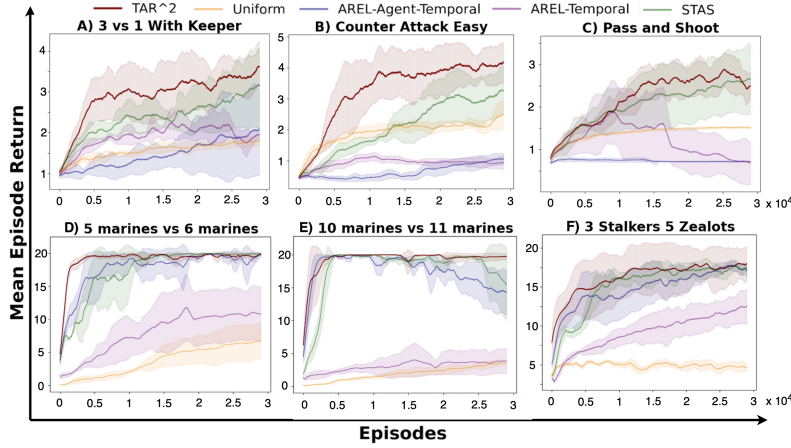


Figure 2: TAR²’s Average Return comparison against baselines on Google Research Football (A-C) and SMACLite (D-F). On SMACLite, it demonstrates improved sample efficiency compared to STAS and converges to a higher average return than the unstable AREL variants. This trend is more pronounced on GRF, where TAR² consistently achieves the highest average return, particularly in ‘Counter Attack Easy’ and ‘Pass and Shoot’ scenarios.

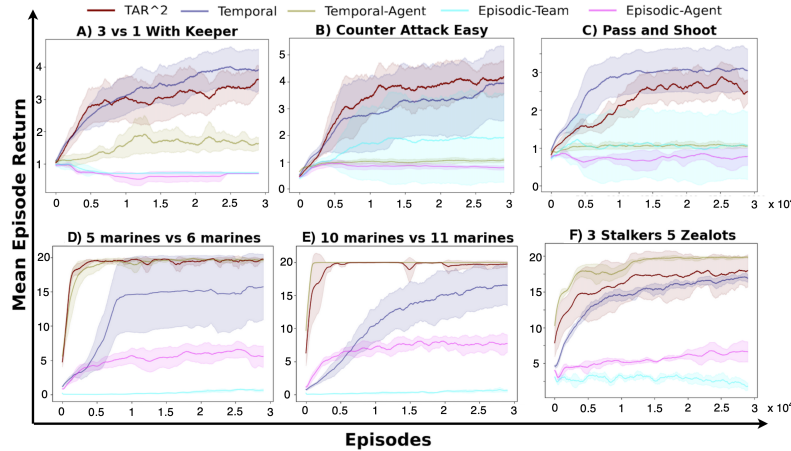


Figure 3: Performance of TAR² relative to oracle rewards on SMACLite (D-F) and Google Research Football (A-C). TAR² enables learning a policy that is competitive with hand-crafted reward functions. TAR²’s performance rivals ‘Temporal-Agent’ in SMACLite and ‘Temporal’ in GRF. It outperforms all other heuristics, demonstrating that a learned credit assignment can be more effective than a manually engineered one.

(RUDDER), using sequence models or retrospectively re-evaluating actions (Arjona-Medina et al., 2019; Liu et al., 2019; Harutyunyan et al., 2019), while others use *intrinsic motivation* to generate dense rewards for exploration (Schäfer et al., 2022). These methods, however, have two key limitations in our context. First, they are designed for single-agent problems and do not address the multi-agent nature of credit assignment. Second, approaches like intrinsic motivation intentionally alter the optimization objective to encourage exploration, whereas our goal is to preserve the original optimal policy. Even methods adapted for MARL, such as AREL (Xiao et al., 2022), address temporal credit for the team as a whole but fail to disentangle individual agent contributions.

4.2 AGENT CREDIT ASSIGNMENT

Agent credit assignment allocates a shared team reward to individual agents, dominated by methods like value function factorization (VDN, QMIX) and counterfactuals (COMA) (Sunehag et al., 2017;

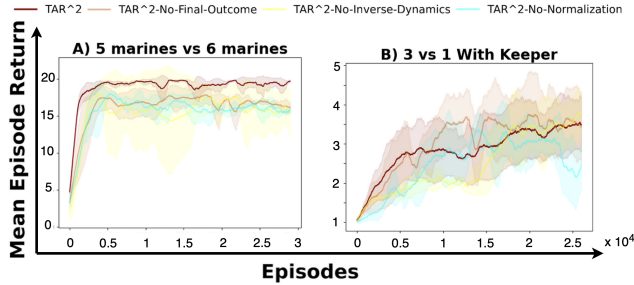


Figure 4: Ablation study of TAR²'s core components. Removing any component degrades performance. 'No-Final-Outcome' increases variance, 'No-Inverse-Dynamics' hinders performance, and 'No-Normalization' is the most detrimental as it violates the policy preservation guarantee.

Rashid et al., 2020; Foerster et al., 2018). Other approaches use Shapley values or attention-based critics (Wang et al., 2020; Freed et al., 2022; Kapoor et al., 2024). A unifying limitation of these methods is their fundamental reliance on dense, per-timestep team rewards, an assumption that fails in the challenging episodic settings we address (Papoudakis et al., 2021). In contrast, TAR² is designed specifically for a single episodic signal.

4.3 JOINT AGENT-TEMPORAL CREDIT ASSIGNMENT

Addressing both credit assignment dimensions is a key frontier in MARL. The most notable prior work, STAS (Chen et al., 2023), tackles this joint problem but its approach is theoretically brittle. It trains a model to directly predict the final shaped rewards, meaning its policy preservation guarantee is conditional on the network's regression accuracy. This reliance on perfect function approximation is the key theoretical fragility that TAR² is designed to overcome, as we detail in Sec 3.

5 EXPERIMENTS

We conduct experiments to answer three core questions: (1) How does TAR² compare to other reward redistribution baselines across diverse and challenging environments? (2) How does its performance compare against oracles with access to privileged reward information? (3) Which of our architectural components are most critical to its success?

5.1 EXPERIMENTAL SETUP

Environments. We evaluate on two challenging benchmarks, SMACLite (Michalski et al., 2023) and Google Research Football (GRF) (Kurach et al., 2020), modifying both to be strictly episodic with a single terminal team reward. Our chosen maps test distinct coordination challenges: SMACLite's *5m vs 6m* and *10m vs 11m* test scalability, while *3s5z* tests coordination between heterogeneous agents. For GRF, we use *3 vs 1 with keeper*, *counterattack easy*, and *pass and shoot* to assess performance across diverse strategies. Further details are in the appendix.

Baselines. For a fair comparison, all methods are built upon the same state-of-the-art MAPPO implementation (Yu et al., 2022). We compare against several credit assignment frameworks that operate on the same episodic team reward. These include a naive Uniform credit baseline; AREL-Temporal (Xiao et al., 2022) for temporal-only assignment; and two strong joint agent-temporal methods, STAS (Chen et al., 2023), the current state-of-the-art, and our adapted AREL-Agent-Temporal. We omit value decomposition methods like QMIX as they rely on Temporal-Difference (TD) updates, which are ill-suited for settings with only a single, delayed episodic reward (Gangwani et al., 2020).

Oracle Baselines. To contextualize TAR²'s performance, we establish oracle baselines using privileged information from the original, dense-reward environments. The original environments provide a dense, per-timestep team reward, which we term Temporal. We then create a factorized version

of this reward that provides a per-timestep, per-agent signal, termed Temporal-Agent. For our main experiments, the standard input for TAR² and all baselines is the Episodic-Team reward, which is the sum of Temporal rewards over an episode. Similarly, Episodic-Agent reward is the sum of Temporal-Agent rewards. This allows us to compare against oracles trained with more information, thereby establishing heuristic performance bounds.

5.2 RESULTS AND DISCUSSION

All learning curves show the mean episode return over 5 random seeds, with shaded areas representing 95% confidence intervals. We use average return as our primary metric to clearly analyze performance gains throughout the entire training process.

Q1: Performance Against Baselines. As shown in Figure 2, TAR² consistently outperforms all baselines in both final average return and sample efficiency. The results empirically validate our core thesis: the structural robustness of TAR² provides a more reliable learning signal than the theoretically brittle designs of prior work. For instance, while STAS is competitive in some SMACLite scenarios, it exhibits higher variance and is unstable in GRF. This aligns with our analysis that its reliance on direct reward regression is fragile. The poor performance of the AREL variants further supports this conclusion. The performance of the Uniform baseline is particularly telling, it fails completely in SMACLite, yet is surprisingly effective in GRF, outperforming more complex methods. This highlights that without a robust theoretical grounding, even advanced models can struggle to beat simple heuristics. TAR²’s stable learning across all challenging environments demonstrates the practical benefit of decoupling credit modeling from constraint satisfaction.

Q2: Contextualizing Performance with Oracle Bounds. Figure 3 contextualizes TAR²’s performance by comparing it against oracles with access to privileged reward signals. The results show that TAR² learns a highly effective credit assignment strategy using only the sparse Episodic-Team reward. In SMACLite, particularly on the homogeneous maps (*5m vs 6m*, *10m vs 11m*), TAR²’s performance is indistinguishable from an oracle trained on perfect, per-agent dense rewards (Temporal-Agent signal). Even on the heterogeneous map (*3s5z*), TAR² learns a near-optimal credit distribution, significantly outperforming weaker oracle signals. In GRF, TAR² is consistently the best-performing non-oracle method and is even superior to all the oracle heuristics in the *Counter Attack Easy* scenario. These results indicate that TAR² learns a credit distribution that is highly competitive with, and at times better than, what can be achieved with hand-crafted, privileged reward functions.

Q3: Ablation Studies of Architectural Components. Our ablation studies (Figure 4) confirm that each component of our design is critical to its success. Removing Final Outcome Conditioning increases learning variance, which is consistent with our analysis in Sec 3.5.1. Ablating the Inverse Dynamics regularizer degrades performance, confirming that grounding the representations in causal actions is crucial (Pathak et al., 2017; Brandfonbrener et al., 2023). Learning less meaningful state representations leads to faulty credit assignment and instability, a known failure mode (Kapoor et al., 2024) due to imprecise credit assignment. The most significant performance collapse occurs when removing our Deterministic Normalization function. This ablation forces the model to suffer from the same fundamental flaw as AREL and STAS: optimizing an objective with no structural policy preservation guarantee. The resulting instability is a direct consequence of this brittle design. Collectively, these studies provide strong evidence that our components work synergistically to reduce variance, promote causal representations, and structurally guarantee policy preservation.

6 LIMITATIONS AND FUTURE WORK

While TAR² establishes a robust framework for episodic credit assignment, several exciting avenues for future work remain. First, the transformer-based architecture may face scalability challenges and could be enhanced by exploring methods like sparse attention or explicit group decomposition to better model agent interactions in massive-scale systems. Second, our framework is currently designed for a single terminal reward; a key next step is to extend TAR² to handle scenarios with multiple sparse rewards within an episode, which would require adapting our formulation. Furthermore, the residual performance variance observed in our results may stem from the implicit exploration encouraged by PBRS (Devlin et al., 2014). A formal exploration-exploitation analysis

could lead to adaptive shaping strategies. A more advanced approach would be to frame this as a bi-level optimization problem, where the reward model is meta-learned to produce shaping signals that directly maximize the downstream performance improvement of the agent policies. Finally, TAR²'s ability to learn from a single, outcome-based signal makes it a prime candidate for training teams of multi-agent Large Language Models (LLMs), a domain where feedback is often sparse.

7 CONCLUSION

We introduced TAR², a method for joint agent-temporal credit assignment designed for structural robustness. By decoupling credit modeling from constraint satisfaction, our approach overcomes the theoretical fragility of prior methods that rely on unreliable reward regression. The method's deterministic normalization step guarantees strict return equivalence by construction, which we prove is equivalent to a valid Potential-Based Reward Shaping (PBRS), ensuring the optimal policy is preserved. Experiments on challenging SMACLite and GRF scenarios show that our approach learns faster and achieves better final performance than state-of-the-art baselines. These results validate our design principle and establish a robust and theoretically-grounded method for credit assignment in complex episodic MARL tasks.

REFERENCES

- Stefano V. Albrecht, Filippos Christianos, and Lukas Schäfer. *Multi-Agent Reinforcement Learning: Foundations and Modern Approaches*. MIT Press, 2024. URL <https://www.marl-book.com>.
- Christopher Amato. (a partial survey of) decentralized, cooperative multi-agent reinforcement learning. *arXiv preprint arXiv:2405.06161*, 2024.
- Jose A Arjona-Medina, Michael Gillhofer, Michael Widrich, Thomas Unterthiner, Johannes Brandstetter, and Sepp Hochreiter. Rudder: Return decomposition for delayed rewards. *Advances in Neural Information Processing Systems*, 32, 2019.
- David Brandfonbrener, Ofir Nachum, and Joan Bruna. Inverse dynamics pretraining learns good representations for multitask imitation, 2023. URL <https://arxiv.org/abs/2305.16985>.
- Sirui Chen, Zhaowei Zhang, Yali Du, and Yaodong Yang. Stas: Spatial-temporal return decomposition for multi-agent reinforcement learning. *ArXiv*, abs/2304.07520, 2023. URL <https://api.semanticscholar.org/CorpusID:258179477>.
- Sam Devlin and Daniel Kudenko. Theoretical considerations of potential-based reward shaping for multi-agent systems. In *Adaptive Agents and Multi-Agent Systems*, 2011. URL <https://api.semanticscholar.org/CorpusID:1116773>.
- Sam Devlin, Logan Yliniemi, Daniel Kudenko, and Kagan Tumer. Potential-based difference rewards for multiagent reinforcement learning. In *Proceedings of the 2014 international conference on Autonomous agents and multi-agent systems*, pp. 165–172, 2014.
- Jakob Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. Counterfactual multi-agent policy gradients. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.
- Benjamin Freed, Aditya Kapoor, Ian Abraham, Jeff Schneider, and Howie Choset. Learning cooperative multi-agent policies with partial reward decoupling. *IEEE Robotics and Automation Letters*, 7(2):890–897, April 2022. ISSN 2377-3774. doi: 10.1109/lra.2021.3135930. URL <http://dx.doi.org/10.1109/LRA.2021.3135930>.
- Tanmay Gangwani, Yuan Zhou, and Jian Peng. Learning guidance rewards with trajectory-space smoothing. *Advances in Neural Information Processing Systems*, 33:822–832, 2020.
- Anna Harutyunyan, Will Dabney, Thomas Mesnard, Mohammad Gheshlaghi Azar, Bilal Piot, Nicolas Heess, Hado P van Hasselt, Gregory Wayne, Satinder Singh, Doina Precup, et al. Hindsight credit assignment. *Advances in neural information processing systems*, 32, 2019.

-
- Aditya Kapoor, Benjamin Freed, Howie Choset, and Jeff Schneider. Assigning credit with partial reward decoupling in multi-agent proximal policy optimization. *arXiv preprint arXiv:2408.04295*, 2024.
- Aleksandar Krnjaic, Raul D. Steleac, Jonathan D. Thomas, Georgios Papoudakis, Lukas Schäfer, Andrew Wing Keung To, Kuan-Ho Lao, Murat Cubuktepe, Matthew Haley, Peter Börsting, and Stefano V. Albrecht. Scalable multi-agent reinforcement learning for warehouse logistics with robotic and human co-workers. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2023.
- Karol Kurach, Anton Raichuk, Piotr Stańczyk, Michał Zajac, Olivier Bachem, Lasse Espeholt, Carlos Riquelme, Damien Vincent, Marcin Michalski, Olivier Bousquet, et al. Google research football: A novel reinforcement learning environment. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pp. 4501–4510, 2020.
- Yang Liu, Yunan Luo, Yuanyi Zhong, Xi Chen, Qiang Liu, and Jian Peng. Sequence modeling of temporal credit assignment for episodic reinforcement learning. *arXiv preprint arXiv:1905.13420*, 2019.
- Ryan Lowe, Aviv Tamar, Jean Harb, OpenAI Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. *Advances in neural information processing systems*, 30, 2017.
- Adam Michalski, Filippos Christianos, and Stefano V. Albrecht. SMACLite: A lightweight environment for multi-agent reinforcement learning. In *AAMAS Workshop on Multiagent Sequential Decision Making Under Uncertainty (MSDM)*, 2023.
- AY Ng. Policy invariance under reward transformations: Theory and application to reward shaping. In *Proceedings of the 16th International Conference on Machine Learning*, pp. 278, 1999.
- Frans A. Oliehoek and Chris Amato. A concise introduction to decentralized pomdps. In *SpringerBriefs in Intelligent Systems*, 2016. URL <https://api.semanticscholar.org/CorpusID:3263887>.
- Georgios Papoudakis, Filippos Christianos, Lukas Schäfer, and Stefano 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.
- Deepak Pathak, Pulkit Agrawal, Alexei A. Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction, 2017. URL <https://arxiv.org/abs/1705.05363>.
- 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.
- Mikayel Samvelyan, Tabish Rashid, Christian Schroeder De Witt, Gregory Farquhar, Nantas Nardelli, Tim GJ Rudner, Chia-Man Hung, Philip HS Torr, Jakob Foerster, and Shimon Whiteson. The starcraft multi-agent challenge. *arXiv preprint arXiv:1902.04043*, 2019.
- Guillaume Sartoretti, Justin Kerr, Yunfei Shi, Glenn Wagner, TK Satish Kumar, Sven Koenig, and Howie Choset. Primal: Pathfinding via reinforcement and imitation multi-agent learning. *IEEE Robotics and Automation Letters*, 4(3):2378–2385, 2019.
- Lukas Schäfer, Filippos Christianos, Josiah P. Hanna, and Stefano V. Albrecht. Decoupled reinforcement learning to stabilise intrinsically-motivated exploration. In *International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, 2022.
- Peter Sunehag, Guy Lever, Audrunas Gruslys, Wojciech Marian Czarnecki, Vinicius Zambaldi, Max Jaderberg, Marc Lanctot, Nicolas Sonnerat, Joel Z Leibo, Karl Tuyls, et al. Value-decomposition networks for cooperative multi-agent learning. *arXiv preprint arXiv:1706.05296*, 2017.

Oriol Vinyals, Igor Babuschkin, Wojciech M. Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David Choi, Richard Powell, Timo Ewalds, Petko Georgiev, et al. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, 575:350 – 354, 2019. URL <https://api.semanticscholar.org/CorpusID:204972004>.

Jianhong Wang, Yuan Zhang, Tae-Kyun Kim, and Yunjie Gu. Shapley q-value: A local reward approach to solve global reward games. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 7285–7292, 2020.

Baicen Xiao, Bhaskar Ramasubramanian, and Radha Poovendran. Agent-temporal attention for reward redistribution in episodic multi-agent reinforcement learning. *arXiv preprint arXiv:2201.04612*, 2022.

Chao Yu, Akash Velu, Eugene Vinitsky, Jiaxuan Gao, Yu Wang, Alexandre Bayen, and Yi Wu. The surprising effectiveness of ppo in cooperative multi-agent games. *Advances in Neural Information Processing Systems*, 35:24611–24624, 2022.

A DETAILED TASK DESCRIPTIONS

SMACLite (Michalski et al., 2023) A computationally efficient variant of StarCraft II (Samvelyan et al., 2019). We experiment on three battle scenarios with varying complexity:

5m_vs_6m & 10m_vs_11m Homogeneous scenarios testing scalability.

3s5z A heterogeneous scenario with 3 Stalkers and 5 Zealots, testing coordination between different unit types.

Each agent’s local observation includes the relative positions, unit types, health, and shield strength of allies and enemies within its field of view, as well as its own status. Agents can move, stop, or attack any visible enemy. The environment provides action masks for valid actions. Each combat scenario lasts for 100 timesteps. The environment’s reward function combines partial rewards for damaging or eliminating enemies, with a maximum possible team return normalized to 20. The repository is available at: <https://github.com/uee-agents/smaclite> (MIT License).

Google Research Football (GRF) (Kurach et al., 2020) A high-fidelity multi-agent football (soccer) simulation. We evaluate on three scenarios:

academy_3_vs_1_with_keeper A basic offensive scenario.

academy_counterattack_easy Tests rapid transitions from defense to offense.

academy_pass_and_shoot_with_keeper Requires precise passing and coordination to score.

The observation space (‘simple115v2’) includes player positions, ball coordinates, velocity vectors, and stamina. The action space includes passing, shooting, dribbling, and tackling. Episodes end after a goal or 200 timesteps. Reward signals are sparse, tied to events like scoring goals or advancing the ball. The repository is available at: <https://github.com/google-research/football> (Apache License 2.0).

B IMPLEMENTATION DETAILS AND HYPERPARAMETERS

The code was run on Lambda Labs deep learning workstation with 2-4 Nvidia RTX 2080 Ti graphics cards. Each training run was run on one single GPU, and required approximately 10 hours.

Hyperparameters used for TAR², STAS, AREL-Temporal, AREL-Agent-Temporal, Uniform and various environment reward configurations that are common to all tasks are shown in Tables 1. The task-specific hyperparameters considered in our grid search for TAR², STAS, AREL-variants in Tables 2, 3 and 4 respectively. Bold values indicate the optimal hyperparameters.

Table 1: Common Hyperparameters for MAPPO algorithms.

common hyperparameters	value
ppo_epochs	15
ppo_batch_size	30
gamma	0.99
max_episodes	30000
max_time_steps	100
rnn_num_layers_v	1
rnn_hidden_v	64
v_value_lr	5e-4
v_weight_decay	0.0
v_hidden_shape	64
grad_clip_critic_v	0.5
value_clip	0.2
data_chunk_length	10
rnn_num_layers_actor	1
rnn_hidden_actor	64
policy_lr	5e-4
policy_weight_decay	0.0
policy_hidden_shape	64
grad_clip_actor	0.5
policy_clip	0.2
entropy_pen	1e-2
gae_lambda	0.95

Table 2: TAR² hyperparameters.

Env. Name	num heads	depth	dropout	comp. dim	batch size	lr	weight decay	inv. dyn. loss coef.	grad clip val.	model upd. freq.	model upd. epochs	policy lr	entropy coef
Google Football	[3, 4]	[3, 4]	[0.0, 0.1, 0.2]	[16, 64, 128]	[32, 64, 128]	[1e-4, 5e-4, 1e-3]	[0.0, 1e-5, 1e-4]	[1e-3, 1e-2, 5e-2]	[0.5, 5.0, 10.0]	[50, 100, 200]	[100, 200, 400]	[5e-4, 1e-3]	[5e-3, 8e-3, 1e-2]
SMACLite	[3, 4]	[3, 4]	[0.0, 0.1, 0.2]	[16, 64, 128]	[32, 64, 128]	[1e-4, 5e-4, 1e-3]	[0.0, 1e-5, 1e-4]	[1e-3, 1e-2, 5e-2]	[0.5, 5.0, 10.0]	[50, 100, 200]	[100, 200, 400]	[5e-4, 1e-3]	[5e-3, 8e-3, 1e-2]

Table 3: STAS hyperparameters.

Env. Name	num heads	depth	dropout	comp. dim	batch size	lr	weight decay	grad clip val.	model upd. freq.	model upd. epochs
Google Football	[3, 4]	[3, 4]	[0.0, 0.1, 0.2]	[16, 64, 128]	[32, 64, 128]	[1e-4, 5e-4, 1e-3]	[0.0, 1e-5, 1e-4]	[0.5, 5.0, 10.0]	[50, 100, 200]	[100, 200, 400]
SMACLite	[3, 4]	[3, 4]	[0.0, 0.1, 0.2]	[16, 64, 128]	[32, 64, 128]	[1e-4, 5e-4, 1e-3]	[0.0, 1e-5, 1e-4]	[0.5, 5.0, 10.0]	[50, 100, 200]	[100, 200, 400]

Table 4: AREL hyperparameters.

Env. Name	num heads	depth	dropout	comp. dim	batch size	lr	weight decay	grad clip val.	model upd. freq.	model upd. epochs
Google Football	[3, 4]	[3, 4]	[0.0, 0.1, 0.2]	[16, 64, 128]	[32, 64, 128]	[1e-4, 5e-4, 1e-3]	[0.0, 1e-5, 1e-4]	[0.5, 5.0, 10.0]	[50, 100, 200]	[100, 200, 400]
SMACLite	[3, 4]	[3, 4]	[0.0, 0.1, 0.2]	[16, 64, 128]	[32, 64, 128]	[1e-4, 5e-4, 1e-3]	[0.0, 1e-5, 1e-4]	[0.5, 5.0, 10.0]	[50, 100, 200]	[100, 200, 400]

C PSEUDOCODE

Our training process is detailed in the following algorithms. Algorithm 1 describes the main on-policy training loop which collects data and updates the MAPPO actor and critic policies. Algorithm 2 describes the periodic, off-policy training of the TAR² reward model.

C.0.1 INVERSE DYNAMICS FOR CAUSAL REPRESENTATION LEARNING

To ensure that the learned representations are grounded in agent behavior, we integrate an auxiliary inverse dynamics task into our framework. The goal of this task is to regularize the shared embeddings by forcing them to encode information about the actions that cause transitions. Our implementation is designed to leverage the rich, contextualized embeddings produced by our main architecture.

Our credit assignment model uses a dual temporal-agent attention network to produce global state-action embeddings for each timestep. These embeddings capture not only the state of the environment but also the interactions between agents over time. To form the input for our inverse dynamics prediction, for each agent i at each timestep t , we create a concatenated vector, $\mathbf{z}_{i,t}$, from three distinct sources:

1. The global state embedding at the current timestep, $\text{embed}(s_t)$.
2. The global state embedding at the next timestep, $\text{embed}(s_{t+1})$.
3. The agent-specific state-action embedding from the *previous* timestep, $\text{embed}(s_{i,t-1}, a_{i,t-1})$, which is the output of the dual attention block for that agent.

This concatenated vector $\mathbf{z}_{i,t}$ is then passed through a multi-layer perceptron (MLP), which we refer to as the inverse dynamics head.

Crucially, the network is not trained to predict the action that was actually executed, but rather the action that the agent’s policy, π_i , predicted at that timestep. The objective is to minimize the cross-entropy between the output of the inverse dynamics head and the action probabilities from the policy network. This encourages consistency between the representations used for credit assignment and the representations used for action selection, ensuring the credit is assigned based on features that are directly relevant to the agent’s decision-making process.

D INTERPRETABILITY AND INSIGHTS

Although our primary focus is on performance and theoretical properties, TAR²’s per-timestep, per-agent reward predictions lend themselves to partial interpretability:

- Agents’ importance at specific timesteps can be visualized by examining $w_t^{\text{temporal}} w_{i,t}^{\text{agent}}$.
- Comparing predicted reward distributions across different episodes can hint at consistent agent roles or strategic pivot points in the trajectory.

However, direct interpretability is challenging in high-dimensional multi-agent environments like SMACLite and Google Football, where the intricate interactions and vast state-action spaces complicate simple visualizations. Additionally, developing a systematic interpretability study would

Algorithm 1 Main Training Loop: MAPPO with TAR² Rewards

```
1: Initialize: Policy nets  $\pi_{\omega_i}$ , critic nets  $V_{\mu_i}$  for each agent  $i = 1..N$ .
2: Initialize: TAR2 reward model  $R_\theta$ , experience buffer  $\mathcal{B} \leftarrow \emptyset$ .
3: Initialize: PopArt parameters for value normalization.
4: while not converged do
5:   Initialize temporary data buffer  $D \leftarrow \emptyset$ .
6:   for  $k = 1$  to  $\text{num\_rollout\_threads}$  do
7:     Initialize actor RNN hidden states  $\mathbf{h}_{0,\pi}$  and critic RNN hidden states  $\mathbf{h}_{0,V}$ .
8:     Initialize empty trajectory storage  $\tau_{\text{storage}} \leftarrow []$ .
9:     for  $t = 0$  to  $T - 1$  do
10:      Get joint observation  $\mathbf{o}_t$ .
11:      for each agent  $i = 1..N$  do
12:        Sample action  $a_{i,t}$  and get next hidden state  $h_{t+1,\pi}^{(i)}$  from policy:
13:         $a_{i,t}, h_{t+1,\pi}^{(i)} \leftarrow \pi_{\omega_i}(\mathbf{o}_t, h_{t,\pi}^{(i)})$ .
14:        Get state value  $v_{i,t}$  and next hidden state  $h_{t+1,V}^{(i)}$  from critic:
15:         $v_{i,t}, h_{t+1,V}^{(i)} \leftarrow V_{\mu_i}(\mathbf{s}_t, h_{t,V}^{(i)})$ , where  $\mathbf{s}_t$  is the centralized state representation.
16:      end for
17:      Execute joint action  $\mathbf{a}_t$ , observe next joint observation  $\mathbf{o}_{t+1}$ .
18:      Store transition  $(\mathbf{o}_t, \mathbf{a}_t, \mathbf{h}_{t+1,\pi}, \mathbf{h}_{t+1,V}, \{v_{i,t}\}_{i=1}^N)$  in  $\tau_{\text{storage}}$ .
19:    end for
20:    Receive the true episodic team reward  $R(s_T)$ .
21:    Store the completed trajectory  $(\tau_{\text{storage}}, R(s_T))$  in the long-term experience buffer  $\mathcal{B}$ .
22:    // — On-Policy Return and Advantage Calculation —
23:    Compute shaped rewards  $\{r_{i,t}\}$  for the trajectory using the TAR2 model  $R_\theta$  and  $R(s_T)$ .
24:    Compute shaped returns  $\{G_{i,t}\}$  for each agent using the shaped rewards  $\{r_{i,t}\}$ .
25:    Update PopArt statistics with the shaped returns  $\{G_{i,t}\}$ .
26:    Normalize returns with PopArt.
27:    De-normalize value estimates  $\{v_{i,t}\}$  using PopArt.
28:    Compute advantage estimates  $\{\hat{A}_{i,t}\}$  for each agent using GAE.
29:    Split trajectory  $\tau$  into chunks of length  $L$ .
30:    for  $l = 0, 1, \dots, T//L$  do
31:       $D = D \cup (\tau[l : l + T], \hat{A}[l : l + L], G[l : l + L], \bar{G}[l : l + L])$ 
32:    end for
33:    Add processed chunks to the data buffer  $D$ .
34:  end for
35:  // — On-Policy Policy and Critic Updates —
36:  for  $e = 1$  to  $\text{ppo\_epochs}$  do
37:    for mini-batch  $b$  sampled from  $D$  do
38:      Update policy parameters  $\omega$  using the MAPPO policy loss on advantage estimates from batch  $b$ .
39:      Update critic parameters  $\mu$  by regressing on the PopArt-normalized shaped returns from batch  $b$ .
40:    end for
41:  end for
42:  // — Off-Policy Reward Model Update —
43:  if training condition met then
44:    Update TAR2 reward model parameters  $\theta$  using Algorithm 2.
45:  end if
46: end while
```

require significant additional methodologies and resources, extending beyond the scope of the current work. While we recognize the importance of interpretability and plan to explore it in future research, our current focus remains on establishing robust performance improvements and theoretical guarantees for reward redistribution.

Algorithm 2 Training the TAR² Reward Model (Off-Policy)

- 1: **Input:** Reward model parameters θ , experience buffer \mathcal{B} , batch size B_r , learning rate α_R .
 - 2: **for** $u = 1$ to $\text{num_reward_updates}$ **do**
 - 3: Sample a batch of trajectories $\{(\tau_j, R(s_T)_j)\}_{j=1}^{B_r}$ from the long-term buffer \mathcal{B} .
 - 4: Initialize loss $\mathcal{L}(\theta) \leftarrow 0$.
 - 5: **for** each trajectory $(\tau, R(s_T))$ in the batch **do**
 - 6: Compute unnormalized scores $\{c_{i,t}\}$ and predicted actions $\{\hat{a}_{i,t}\}$ from the reward model $R_\theta(\tau)$.
 - 7: $\mathcal{L}_{\text{reg}} = \left(\log R(s_T) - \sum_{t,i} c_{i,t} \right)^2$.
 - 8: $\mathcal{L}_{\text{ID}} = - \sum_{t,i} \log \pi_{\text{ID}}(a_{i,t} | \hat{a}_{i,t})$.
 - 9: $\mathcal{L}(\theta) += \mathcal{L}_{\text{reg}} + \lambda \mathcal{L}_{\text{ID}}$.
 - 10: **end for**
 - 11: Update θ using Adam optimizer: $\theta \leftarrow \theta - \alpha_R \nabla_\theta \mathcal{L}(\theta)$.
 - 12: **end for**
-