Know Your Task, Learn It Right: Task-Aware Optimistic Value Learning for Multi-Task Multi-Agent Reinforcement Learning

Chang Liu
University of Florida
changliu2@ufl.edu

Mengyang Li*
Tianjin Normal University
limengyang@tjnu.edu.cn

Abstract

Cooperative multi-agent reinforcement learning over heterogeneous episodes requires both identifying which latent class an episode belongs to and adapting the learning algorithm to that identity. Recent trajectory-class-aware methods address the first half of this problem by clustering trajectories into latent classes and feeding the class label to the agent policy. Class-dependent learning dynamics also motivate adapting the Bellman update and the credit assignment performed by the value mixer. We show empirically that, even when classes are recovered accurately, the Bellman dynamics across classes diverge sharply, with simple classes saturating early and harder classes sustaining large temporal-difference residuals and large cross-agent value disagreement. Building on this observation, we propose Know Your Task, Learn It Right (KYT), a class-aware extension of value factorization that injects per-class learning signal into the Bellman update itself. KYT maintains a lightweight tracker that summarizes the per-class TD-error and cross-agent value spread without any extra network, and uses it to add a bounded, class-relative exploration bonus to the Bellman target. A class-conditioned mixer further modulates credit assignment through a softplus-gated affine layer that preserves the Individual-Global-Max (IGM) property, so that decentralized greedy execution stays consistent with the centrally trained value for any predicted class label. Across StarCraft II micromanagement benchmarks including SMACv2 and the unit-combination suite SurComb, KYT outperforms strong class-aware and exploration-aware baselines, with the largest gains concentrated on the hardest classes.

1 Introduction

Cooperative multi-agent reinforcement learning (MARL) is often trained on a heterogeneous mixture of episodes. Within a single environment, different enemy compositions, spawn configurations, or sub-goal structures induce latent classes of trajectories that call for different team behavior, and the team must learn a single conditional policy that performs well on all of them without observing the class at execution time [Sunehag et al., 2018, Rashid et al., 2020b, Wang et al., 2021a, Foerster et al., 2018, Lowe et al., 2017, Yu et al., 2022, Liu et al., 2023, Na et al., 2025]. The difficulty is twofold. The team must first infer which class an episode belongs to from short observation prefixes, and then act in a way that is appropriate for that class. A sequence of recent works [Wang et al., 2020a, 2021b, Yuan et al., 2022, Jeon et al., 2022, Zheng et al., 2021, Na et al., 2025] has addressed class identification by clustering trajectories or roles into a small set of latent classes, and feeding the class label or its embedding into the agent policy as an auxiliary input. These methods make accurate class identification practical at scale.

*Corresponding author.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

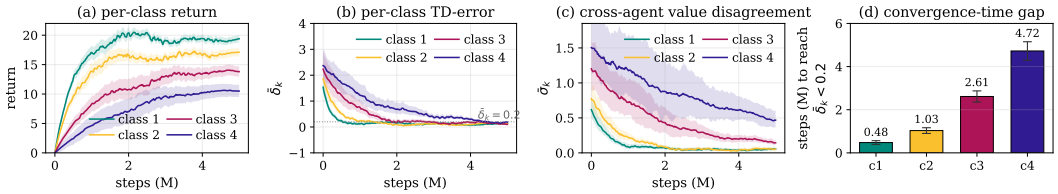


Figure 1: Per-class learning dynamics on SurComb4: (a) team return, (b) TD-error, (c) cross-agent disagreement, and (d) steps to the TD-error threshold of 0.2.

Class identity can also inform value learning. Standard class-aware value-factorization pipelines use a shared temporal-difference update rule and mixer across classes. Their learning dynamics, however, can differ: easy classes saturate early, while harder classes benefit from continued exploration and class-dependent value updates. Adapting the learning signal to these differences can support progress across the full class distribution.

We examine per-class learning dynamics on SurComb4 using a representative trajectory-class-aware learner [Na et al., 2025]. Figure 1 shows that returns separate into easy and hard classes within the first quarter of training. The easiest class reaches an absolute TD-error below 0.2 within 0.5M environment steps, while the hardest requires more than 4.5M. Cross-agent value disagreement, measured as the variance of agent utilities Q^i on the same global state, decreases rapidly on easy classes and remains high on hard ones. Its evolution closely follows the TD-error, suggesting that disagreement is informative about remaining learning progress. Class-dependent dynamics also motivate difficulty-aware training in supervised learning [Li et al., 2023, 2025, 2026]. Here we examine their role in the Bellman update for cooperative MARL.

These observations motivate Know Your Task, Learn It Right (KYT), a class-aware extension of value factorization that uses class identity to guide the Bellman update and credit assignment. A parameter-free tracker maintains three scalars per class: exponential moving averages of TD-error magnitude, cross-agent value spread, and policy entropy. The disagreement signal determines a bounded, class-specific optimistic bonus that decreases as disagreement decays. A class-conditioned mixer applies a softplus-gated affine modulation to the factorized value, preserving the Individual-Global-Max (IGM) property for decentralized greedy execution [Wang et al., 2021a].

Our contributions are:

- We characterize per-class learning dynamics in class-aware MARL: even when classes are recovered accurately, per-class Bellman dynamics differ by an order of magnitude in convergence time, and cross-agent value disagreement tracks this gap closely.
- We propose KYT, a class-aware Bellman update with two mechanisms, an optimistic target driven by per-class disagreement and a class-conditioned mixer with softplus gating, both of which preserve the IGM property and add about 200 learnable parameters.
- We evaluate KYT on ten benchmarks from SMACv2, SurComb, and the original SMAC against seven baselines and show consistent improvements, with the largest gains concentrated on the hardest classes.

2 Preliminaries

2.1 Cooperative MARL and value factorization

We consider a decentralized partially observable Markov decision process [Oliehoek and Amato, 2016] defined by a tuple $\langle \mathcal{N}, \mathcal{S}, \mathcal{A}, P, R, \Omega, O, \gamma \rangle$ with n agents indexed by $\mathcal{N} = \{1, \dots, n\}$, joint action $\mathbf{a} = (a^1, \dots, a^n) \in \mathcal{A}^n$, partial observations $o^i \sim O(s, i)$, transition kernel $P(s' \mid s, \mathbf{a})$, shared reward $R(s, \mathbf{a})$, and discount γ. Each agent conditions on its action-observation history $\tau_t^i = (o_1^i, a_1^i, \dots, o_t^i)$. Building on deep Q-learning [Mnih et al., 2015], value factorization methods [Sunehag et al., 2018, Rashid et al., 2020b, Son et al., 2019, Wang et al., 2021a, Peng et al., 2021] decompose the joint action-value $Q_{tot}(\mathbf{s}, \mathbf{a})$ as a function of per-agent utilities $Q^i(\tau^i, a^i)$ through a learnable mixer that satisfies the Individual-Global-Max (IGM) condition $\arg\max_{\mathbf{a}} Q_{tot}(\mathbf{s}, \mathbf{a}) = \big(\arg\max_{a^i} Q^i(\tau^i, a^i)\big)_{i=1}^n$. IGM is what makes centralized training compatible with decentralized

execution: at execution each agent acts greedily on its own utility, without communication or access to the global state, and IGM guarantees that the resulting joint action still maximizes Q_{tot}. This consistency links the centralized value to the executed policy.

2.2 Latent-class-structured cooperative MARL

Following the convention in recent class-aware MARL [Wang et al., 2021b, Yuan et al., 2022, Na et al., 2025], the training distribution is a mixture indexed by an unobserved variable $z \in \mathcal{Z}$, with class-conditional transition and reward $P(\cdot \mid s, \mathbf{a}; z), R(s, \mathbf{a}; z)$. The team trains a single policy without access to z at training or execution time, and evaluation reports per-class and average return. Because z is unobserved, the policy must internally infer it from short observation prefixes. In this paper z indexes latent trajectory classes within a single environment with a fixed set of agents, with training and evaluation over the same class set. Cross-task generalization considers tasks that vary in the number or type of agents and evaluates policies on unseen tasks [Zhang et al., 2023, Liu et al., 2025, Jeon et al., 2026]. The formulation also overlaps with the meta-RL setup that infers a task variable from a context window [Rakelly et al., 2019]. Here, class inference takes place within the cooperative MARL setting described above.

2.3 Trajectory-class identification

A common approach to recovering class identity in this setting is unsupervised trajectory clustering. A learned encoder maps an action-observation history to a discrete or continuous code, and clustering on this code yields a class label $\hat{k} \in \{1, \ldots, n_{cl}\}$ that serves as a proxy for z. Concrete realizations include role decomposition [Wang et al., 2021b, 2020a], latent sub-task assignment [Yuan et al., 2022], and quantized trajectory embeddings with centroid initialization [Na et al., 2025]. We denote the predicted class label by $\hat{k}$ and the per-agent class representation by g^i.

2.4 Optimism and disagreement-based exploration

Single-agent reinforcement learning has a long tradition of using optimism in the face of uncertainty as an exploration principle, with bonuses derived from visit counts [Strehl and Littman, 2008], prediction-error novelty [Pathak et al., 2017, Burda et al., 2019], ensemble disagreement [Osband et al., 2016, Pathak et al., 2019], or learned directed-exploration policies that combine episodic and lifelong novelty [Badia et al., 2020]. In a cooperative MARL setting, the team naturally maintains n agent-level utilities that are aggregated by the mixer. The variance of these utilities at the same global state, which we refer to as cross-agent value disagreement, is available at every gradient step from the existing agent networks. It combines learning-related variation with persistent differences between agent roles (Appendix A.2). KYT uses the within-benchmark ranking of classes by disagreement as an indicator of remaining learning progress (Appendix C.2). We use this class-level disagreement signal in Section 3 as the basis of a class-relative exploration bonus.

3 Method

3.1 Overview

KYT uses a predicted class label $\hat{k} \in \{1, \ldots, n_{cl}\}$ for each training transition and a per-agent class representation g^i supplied to the agent network. Figure 2 illustrates its three components: a per-class Bellman dynamics tracker ψ_k that summarizes the recent learning signal of every class (Section 3.2), an optimistic Bellman target that uses ψ_k to scale a class-specific exploration bonus (Section 3.3), and a class-conditioned mixer that modulates credit assignment as a function of $\hat{k}$ (Section 3.4). The full training objective is given in Section 3.5.

3.2 Per-class Bellman dynamics tracker

For each class $k \in \{1, \ldots, n_{cl}\}$ we maintain three exponential moving averages (EMAs) updated only on transitions that the identification module assigns to k. Let $\delta^{(T)} = r + \gamma \max_{\mathbf{a}'} Q^{\theta^-}_{tot}(\mathbf{s}', \mathbf{a}') - Q^{\theta}_{tot}(\mathbf{s}, \mathbf{a})$ denote the standard TD-error at training step T and let $\eta \in (0, 1)$ be the EMA rate, so

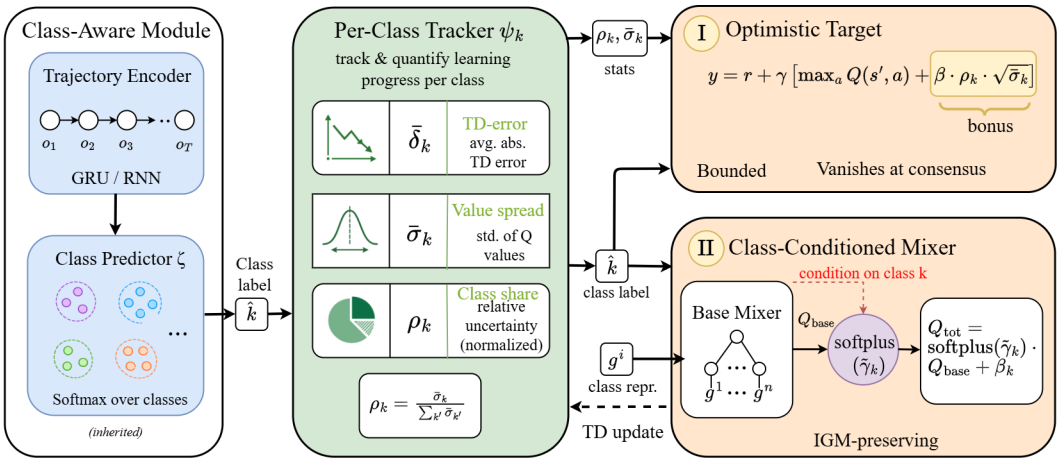


Figure 2: KYT architecture: class representations and labels guide a per-class dynamics tracker, an optimistic Bellman target, and an IGM-preserving class-conditioned mixer.

that the tracker's effective memory is roughly $1/\eta$ updates. The tracker $\psi_k = (\bar{\delta}_k, \bar{\sigma}_k, \bar{H}_k)$ is

$$\bar{\delta}_k(T) = (1-\eta)\,\bar{\delta}_k(T-1) + \eta \cdot \left|\delta^{(T)}\right|\Big|_k, \tag{1}$$

$$\bar{\sigma}_k(T) = (1-\eta)\,\bar{\sigma}_k(T-1) + \eta \cdot \mathrm{Var}_i\left[Q^i_\theta(\tau^i, a^i)\right]\Big|_k, \tag{2}$$

$$\bar{H}_k(T) = (1-\eta)\,\bar{H}_k(T-1) + \eta \cdot H\left[\pi_\theta(\cdot \mid o^i, g^i)\right]\Big|_k. \tag{3}$$

The tracker is a vector of three scalars per class and contains no learnable parameters. The bonus below consumes $\bar{\sigma}_k$; $\bar{\delta}_k$ and $\bar{H}_k$ serve as diagnostics of learning dynamics (Figure 3). To convert the raw disagreement into a relative weight, we define

$$\rho_k(T) = \frac{\bar{\sigma}_k(T)}{\sum_{k'=1}^{n_{cl}} \bar{\sigma}_{k'}(T) + \epsilon}, \tag{4}$$

which can be read as the share of total cross-agent disagreement that class k is currently responsible for. On the benchmarks we test, the ordering of classes by ρ_k agrees with their ordering by per-class return (Figure 3c; Appendix C.2). We use ρ_k rather than $\bar{\sigma}_k$ as the allocation weight because ρ_k is invariant to a global rescaling of Q, so the distribution of the bonus across classes does not drift with the absolute scale of the values; the magnitude of the bonus introduced below scales with $\sqrt{\bar{\sigma}_k}$, i.e., linearly with the scale of Q, like the target to which it is added.

3.3 Task-aware optimistic Bellman target

The first mechanism uses ψ_k to add a class-specific exploration bonus directly to the bootstrapped return. For a transition assigned to class $\hat{k}$ we set

$$y_{\hat{k}} = r + \gamma\left[\max_{\mathbf{a}'} Q^{\theta^-}_{tot}(\mathbf{s}', \mathbf{a}'; g') + \beta\,\rho_{\hat{k}}(T)\,\sqrt{\bar{\sigma}_{\hat{k}}(T)}\right], \tag{5}$$

with $\beta \geq 0$ a single global coefficient. The bonus has three properties. First, it is bounded by $\beta\sqrt{\bar{\sigma}_{\max}}$ since $\rho_k \leq 1$, which relates its magnitude to the largest current class-level disagreement. Second, it shrinks as disagreement on a class decays. When agents play structurally different roles, $\bar{\sigma}_k$ plateaus at a positive floor instead of vanishing, and the bonus settles at a small, bounded, class-dependent constant rather than at zero (Proposition 1; Appendix A.2). Third, the bonus is class-relative because ρ_k shifts mass to whichever class currently shows the largest disagreement; this means easy classes that have already saturated receive little bonus even if β is held constant.

The Bellman loss is then

$$\mathcal{L}_{TD}(\theta, \omega) = \mathbb{E}_{(\mathbf{s},\mathbf{a},r,\mathbf{s}')\sim\mathcal{D}}\left[\left(y_{\hat{k}} - Q_{tot}(\mathbf{s}, \mathbf{a}; \hat{k}, g)\right)^2\right], \tag{6}$$

with $\hat{k}$ supplied by the identification module and Q_{tot} given by the class-conditioned mixer of Equation 7.

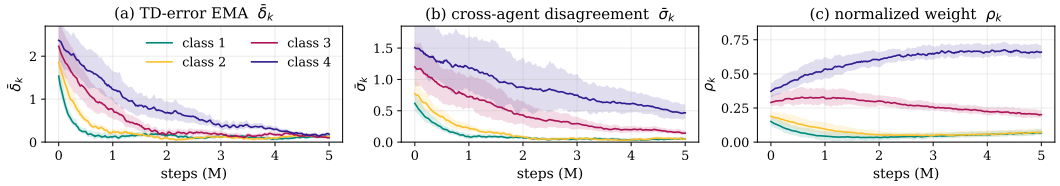


Figure 3: Per-class tracker on SurComb4: (a) TD-error EMA $\bar{\delta}_k$, (b) disagreement $\bar{\sigma}_k$, and (c) normalized relative weight ρ_k. The weights shift toward harder classes as training progresses.

3.4 Class-conditioned mixer

The second mechanism modifies credit assignment so that it can vary with the class label. Let $Q_{tot}^{base}(\mathbf{s}, \mathbf{a})$ denote the output of an existing factorization mixer such as QMIX or QPLEX. We apply a per-class affine modulation on top, in the spirit of feature-wise linear modulation [Perez et al., 2018] and lighter than hypernetwork-generated mixer weights [Ha et al., 2017]:

$$Q_{tot}(\mathbf{s}, \mathbf{a}; \hat{k}) = \mathrm{softplus}(\tilde{\gamma}_{\hat{k}}) \cdot Q_{tot}^{base}(\mathbf{s}, \mathbf{a}) + \beta_{\hat{k}}^{mix}, \qquad (\tilde{\gamma}_k, \beta_k^{mix}) = \mathrm{MLP}_\omega\big(\mathrm{onehot}(k)\big). \quad (7)$$

The MLP ω has a single hidden layer of width 32, so the mixer adds about 200 parameters in total for $n_{cl} = 4$ (Appendix B). Because $\mathrm{softplus}(\cdot) > 0$ for every input, the per-class scaling is strictly positive. Strict positivity implies that for every k the global $\arg\max$ over joint actions is preserved under the per-class modulation, so the IGM condition that the underlying mixer satisfies is still satisfied after Equation 7. The bias β_k^{mix} shifts the absolute level of Q_{tot} across classes but does not affect the $\arg\max$. Together this makes the modulation a strictly monotone, IGM-preserving correction to the base factorization.

3.5 Overall objective

KYT trains the joint set of parameters $(\theta, \omega, \zeta, \phi, e)$, where θ denotes the agent and base mixer networks, ω the gating MLP of Equation 7, ζ the class predictor of the identification module, and (ϕ, e) its trajectory encoder and codebook. The total loss is

$$\mathcal{L}(\theta, \omega, \zeta, \phi, e) = \mathcal{L}_{TD}(\theta, \omega) + \lambda_\zeta \mathcal{L}_{cls}(\zeta) + \lambda_{VQ} \mathcal{L}_{VQ}(\phi, e), \quad (8)$$

with $\mathcal{L}_{TD}$ from Equation 6. The classification loss $\mathcal{L}_{cls}$ and the codebook loss $\mathcal{L}_{VQ}$ follow the trajectory identification objective of Na et al. [2025]. The tracker ψ_k updates after each gradient step using the freshly computed batch quantities; it is decoupled from the loss and contributes no gradient.

3.6 Theoretical properties

We summarize the bonus bounds, limiting behavior, and IGM preservation properties of KYT. The proof is given in Appendix A.1.

Proposition 1 (Bounded bonus, limiting behavior, IGM preservation)**.** For every $\beta \geq 0$ and every step T, the optimistic correction in Equation 5 satisfies $0 \leq \beta\, \rho_{\hat{k}}(T) \sqrt{\bar{\sigma}_{\hat{k}}(T)} \leq \beta \sqrt{\bar{\sigma}_{\max}(T)}$, with $\bar{\sigma}_{\max}(T) = \max_k \bar{\sigma}_k(T)$. If for every class $\bar{\sigma}_k(T) \to \bar{\sigma}_k^\infty \geq 0$ as $T \to \infty$, then the correction converges to the class-dependent constant $\beta\, \rho_k^\infty \sqrt{\bar{\sigma}_k^\infty} \leq \beta \sqrt{\bar{\sigma}_{\max}^\infty}$, with $\rho_k^\infty = \bar{\sigma}_k^\infty / (\sum_{k'} \bar{\sigma}_{k'}^\infty + \epsilon)$; in particular, if $\bar{\sigma}_k^\infty = 0$ for every class, $y_{\hat{k}}$ converges to the standard one-step Bellman target. Moreover, for every class label $\hat{k}$, the modulated mixer in Equation 7 preserves IGM whenever the base mixer does.

The bound relates the optimistic correction to the scale of the current disagreement statistics. The limit is stated for a general $\bar{\sigma}_k^\infty$ because disagreement need not vanish: on maps with heterogeneous unit types agents differ by role, and $\bar{\sigma}_k$ plateaus at a positive floor (Appendix A.2). When the floors are equal across classes the residual bonus is a uniform constant c, which in the infinite-horizon discounted setting shifts Q_{tot} by the state-independent offset $\gamma c/(1-\gamma)$ and leaves the greedy policy unchanged; when they differ it is a bounded class-dependent offset. In episodic tasks a persistent per-step bonus is not exactly policy-neutral, since states closer to termination accumulate less of it;

on MMM2, which has the highest floor we measure, the realized per-step residual is about 0.09. The IGM clause follows from strict positivity of softplus, which makes the per-class affine map strictly increasing and hence $\arg\max$-preserving.

Feedback between the tracker and target. The bonus depends on $\bar{\sigma}_k$, which is computed from the current Q and feeds into the next regression target. Two design choices support smooth updates in this feedback loop. First, with $\eta = 0.01$ the tracker averages over roughly the last 100 gradient steps, so $\bar{\sigma}_k$ varies on a slower time-scale than the regression target and behaves as a near-constant within an update window. Second, the relative weight ρ_k is a share-based normalization across classes, so a uniform rescaling of disagreement across all classes does not change how the bonus is distributed. In Figure 5c the bonus on individual classes decreases from a small number of training steps onward until it reaches its floor, with stable observed behavior across the ten benchmarks.

4 Experiments

4.1 Setup

We evaluate on three groups of cooperative multi-agent benchmarks. SMACv2 [Ellis et al., 2023] is the recent stochastic version of StarCraft II micromanagement; we use the protoss5_vs_5 (p5_vs_5) and terran5_vs_5 (t5_vs_5) maps. SurComb [Na et al., 2025] is a sparse-reward unit-combination suite built on the SMAC engine, with explicit structure inducing several trajectory classes; we use SurComb3, SurComb4 and the harder reSurComb3, reSurComb4 variants. The original SMAC [Samvelyan et al., 2019] contributes single-task diagnostics on MMM2, 6h_vs_8z, 5m_vs_6m, and 1c3s5z. We compare KYT against seven baselines on all ten benchmarks: QMIX, QPLEX, RODE, LDSA, MASER, EMC, and TRAMA [Na et al., 2025]. LAGMA [Liu et al., 2023] was run only on the six benchmarks of Figure 4; we report it there and in Appendix C.7. KYT uses the trajectory encoder and agent backbone of TRAMA [Na et al., 2025], with the proposed target and mixer modulation. We refer to the shared configuration without these two components as the base learner in the ablation and control experiments. Each method is run for $5{\times}10^6$ environment steps with 8 random seeds; we report the mean and the half-width of a 95% percentile bootstrap confidence interval (10^4 resamples) and, for the comparison with TRAMA, the probability of improvement, as recommended by Agarwal et al. [2021]. For SurComb we report team return, i.e., the sum over the five allied agents; all curves use the same convention, so their endpoints coincide with the tables. Figure 1 uses TRAMA as the representative learner for the per-class dynamics analysis. Per-class results are aligned across seeds by enemy composition and ordered by baseline difficulty (Appendix B). SurComb4 evaluation samples its four classes uniformly. The optimism coefficient is fixed to $\beta = 0.5$ across all benchmarks, the EMA rate to $\eta = 0.01$, and the number of classes to $n_{cl} = 4$ on every class-structured suite.[2] None of these is tuned per environment.

4.2 Main results

Table 1 reports final performance at the end of the 5M-step budget. KYT attains the highest mean on all ten benchmarks. The separation is clearest on SurComb, where the 95% CIs of KYT and TRAMA do not overlap on any of the four suites. On SurComb3, the easiest of them, TRAMA already performs well (16.30) and KYT adds a moderate improvement (17.50). As the suites become sparser and harder the gap widens: on reSurComb4 TRAMA reaches a return of 10.70 while KYT reaches 12.30, a 15% relative improvement. The pattern in Table 1 supports adapting the value update to class-dependent learning dynamics, particularly when classes differ substantially in difficulty.

The aggregate gain conceals a strong class dependence. Table 2 decomposes the final SurComb4 return by class: the improvement grows monotonically with class difficulty, from $+0.1$ on the easiest class to $+2.7$ on the hardest, which matches the allocation of the bonus in Figure 5c and of the mixer gates in Figure 6c. Averaging over the final 500k steps also shows the improvement (16.09 ± 0.39 vs. 15.08 ± 0.45 on SurComb4, 12.01 ± 0.52 vs. 10.60 ± 0.57 on reSurComb4; Appendix C.1).

[2] All class-conditioned baselines are retrained with $n_{cl} = 4$ under a common class budget. The original TRAMA configuration uses 3 classes on 1c3s5z, 5m_vs_6m and MMM2; 4 on 6h_vs_8z; 6 on reSurComb3; and 8 on reSurComb4, p5_vs_5 and t5_vs_5. Its results here correspond to our re-runs with the shared budget.

Table 1: Final performance at five million steps: win rate for SMACv2/SMAC and team return for SurComb/reSurComb. Mean $\pm$ 95% bootstrap CI half-width over eight seeds; best means in bold.

Task	QMIX	QPLEX	RODE	LDSA	MASER	EMC	TRAMA	**KYT (ours)**
p5_vs_5	0.54±0.05	0.60±0.05	0.65±0.06	0.68±0.04	0.61±0.07	0.66±0.04	0.73±0.03	**0.77±0.03**
t5_vs_5	0.48±0.06	0.53±0.04	0.58±0.05	0.61±0.04	0.54±0.07	0.59±0.05	0.66±0.04	**0.70±0.03**
SurComb3	13.20±0.71	14.50±0.58	14.80±0.49	15.40±0.62	14.20±0.78	15.20±0.43	16.30±0.41	**17.50±0.38**
SurComb4	10.60±0.83	11.80±0.69	13.30±0.54	14.00±0.51	12.70±0.74	13.60±0.62	15.20±0.47	**16.30±0.42**
reSurComb3	7.90±0.79	9.10±0.66	10.50±0.81	11.20±0.57	10.00±0.73	10.80±0.64	12.30±0.49	**13.60±0.51**
reSurComb4	5.70±0.91	7.20±0.76	8.50±0.69	9.30±0.78	8.20±0.85	8.90±0.71	10.70±0.59	**12.30±0.54**
MMM2	0.65±0.05	0.73±0.04	0.78±0.04	0.81±0.03	0.77±0.05	0.82±0.03	0.87±0.03	**0.92±0.02**
6h_vs_8z	0.43±0.06	0.51±0.05	0.59±0.06	0.64±0.04	0.57±0.07	0.65±0.05	0.73±0.04	**0.79±0.03**
5m_vs_6m	0.68±0.04	0.73±0.04	0.76±0.03	0.78±0.03	0.75±0.04	0.79±0.03	0.83±0.02	**0.86±0.02**
1c3s5z	0.91±0.02	0.94±0.01	0.95±0.01	0.95±0.01	0.94±0.02	0.96±0.01	0.96±0.01	**0.97±0.01**

Table 2: Per-class team return on SurComb4, ordered from easiest to hardest. Mean $\pm$ 95% bootstrap CI half-width over eight seeds; uniform class averages match Table 1.

	Class 1 (easiest)	Class 2	Class 3	Class 4 (hardest)	Mean
TRAMA	19.4±0.62	17.1±0.71	13.8±0.88	10.5±1.05	15.20
KYT	19.5±0.59	17.5±0.70	15.0±0.82	13.2±0.97	16.30
Δ	+0.1	+0.4	+1.2	+2.7	+1.10

With uniform class sampling, classes 3 and 4 account for approximately 89% of the aggregate improvement on SurComb4. The return gap between the easiest and hardest classes decreases from 8.9 to 6.3, while the easiest class retains its high return. Thus, the increase in mean performance is accompanied by more balanced outcomes across classes. This distribution of gains is consistent with allocating larger bonuses to classes that exhibit greater value disagreement.

On the stochastic SMACv2 maps, KYT improves the mean win rate over TRAMA by four percentage points on both p5_vs_5 and t5_vs_5, with overlapping CIs. On the single-task SMAC maps, which we use as a diagnostic for the case where class structure is weak, KYT provides modest improvements. The CIs touch or overlap on MMM2 (0.92 vs. 0.87) and 6h_vs_8z (0.79 vs. 0.73), where the probability of improvement over TRAMA is 0.91 and 0.88 respectively, and both methods saturate on 1c3s5z.

4.3 Bellman dynamics evidence

The motivation for KYT is that per-class Bellman dynamics differ sharply. Figure 5 verifies that the proposed mechanisms act on this gap in the way the design intends. The TD-error EMA $\bar{\delta}_k$ in panel (a) drops faster on every class under KYT than under TRAMA, with the largest acceleration on classes 3 and 4, which are the hardest. The cross-agent disagreement $\bar{\sigma}_k$ in panel (b) follows the same pattern; this supports the role of class-aware updates in reducing value disagreement. Panel (c) shows the realized exploration bonus $\beta\,\rho_k\,\sqrt{\bar{\sigma}_k}$ as a function of training step. The bonus is class-relative by construction: the bonus on the easiest class shrinks first, the bonus on the hardest class stays high until late in training, and at convergence all four classes receive a small bonus. This is the qualitative shape of an automatic class-difficulty curriculum that the method discovers without any external scheduling.

The three panels describe complementary aspects of the learning process. The TD residual measures prediction error, disagreement summarizes variation among agent utilities, and the bonus shows how the tracked signal enters the regression target. Their joint evolution connects class-level statistics with the updates applied during training. The larger changes on classes 3 and 4 are consistent with their stronger return improvements in Table 2, linking the training dynamics to the final evaluation outcomes.

4.4 Ablation

Table 3 isolates the contribution of each KYT component on SurComb4 and reSurComb4. Removing the optimistic target hurts more than removing the class-conditioned mixer, which is consistent with the observation in Figure 5 that the optimism bonus reaches into the hard classes where the bot-

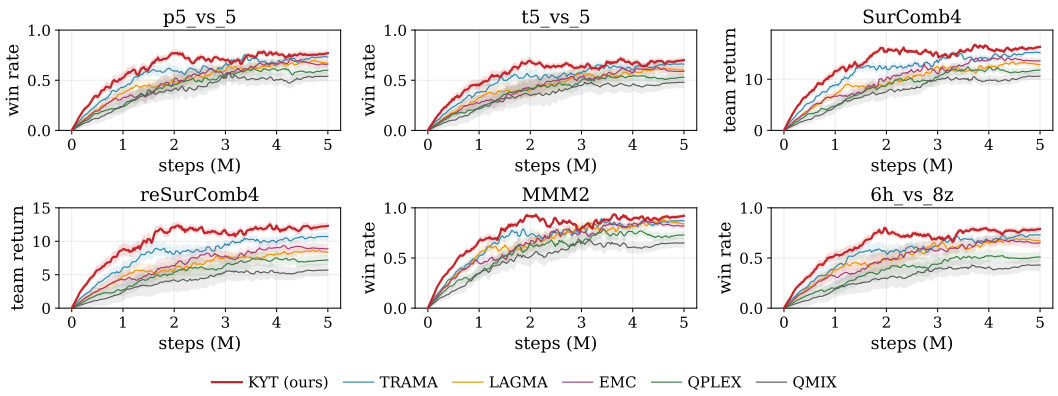


Figure 4: Learning curves on six representative benchmarks. Shaded bands denote 95% bootstrap CIs over eight independent random seeds. SurComb panels report team return summed over five agents; endpoints correspond to Table 1.

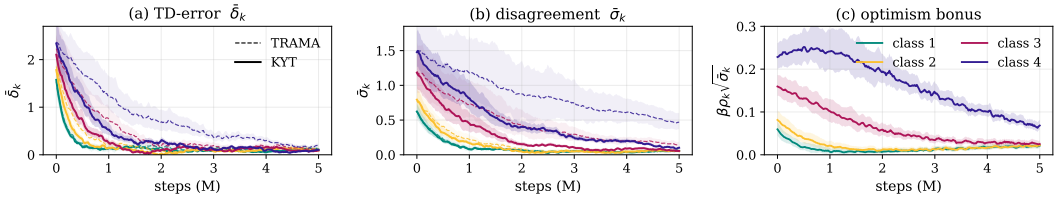


Figure 5: Bellman dynamics on SurComb4: (a) TD-error and (b) disagreement under KYT (solid) and TRAMA (dashed); (c) per-class optimism bonuses. The largest improvements occur on the hardest trajectory classes.

tleneck is. Combining both components improves over either alone, supporting their complementary roles: the optimistic target reshapes the regression target while the conditioned mixer reshapes the function class. Table 3 also reports four control variants, and a fifth (PER-TD) is reported in Appendix C. Replacing the adaptive ρ_k with a uniform $1/n_{cl}$ yields smaller gains over the base learner, supporting the contribution of adaptive per-class weighting. Replacing $\bar{\sigma}_k$ with a fixed scalar of 1.0 also degrades performance, which indicates that the disagreement signal carries useful information for allocating the exploration bonus. Two simple data-level baselines, per-class loss reweighting (multiply each transition's TD loss by $\rho_{\hat{k}}$) and per-class learning rate scheduling (scale Adam's lr by $\rho_{\hat{k}}$), improve on the base learner, while KYT attains higher mean returns, indicating that adapting the Bellman target outperforms distributional reweighting at matched compute. Additional data-level baselines and a robustness study to identification noise are reported in Appendix C.

The component gains also vary with the evaluation setting. Relative to the base learner, the optimistic target improves mean return by 0.8 on SurComb4 and 1.2 on reSurComb4; the mixer contributes 0.5 and 0.7, respectively. Full KYT achieves gains of 1.1 and 1.6, combining target adaptation with class-dependent value scaling. Both components therefore contribute to the improvements observed across these two suites.

4.5 Sensitivity and qualitative analysis

Figure 6 combines four diagnostic panels into a single row. Panels (a) and (b) sweep the two main hyperparameters of the method on SurComb4. KYT is robust to the optimism coefficient β over an order of magnitude around the chosen value of 0.5. Setting $\beta = 0$ removes the bonus from Equation 5 while leaving the class-conditioned mixer in place, so it reproduces the mixer-only variant of Table 3 (15.7 ± 0.48) with a $+0.5$ improvement over the base learner. The sweep over $\beta \in [0.1, 1.0]$ measures the contribution of the optimism term. KYT also improves over TRAMA for every number of classes n_{cl} we tested, with the peak at $n_{cl} = 4$ matching the underlying structure of SurComb4. The EMA rate is similarly uncritical: $\eta \in \{0.003, 0.01, 0.03\}$ gives final returns of 16.18, 16.30, and 16.07, within noise of one another (Appendix C.6). Panel (c) plots the learned mixer gates

Table 3: Ablation on SurComb4 and reSurComb4. Final team return, mean $\pm$ 95% bootstrap CI half-width over eight seeds.

Variant	SurComb4	reSurComb4
Base learner	15.2±.47	10.7±.59
Optimistic target only	16.0±.43	11.9±.55
Class-conditioned mixer only	15.7±.48	11.4±.62
Full KYT	**16.3±.42**	**12.3±.54**
full KYT, uniform $\rho_k = 1/n_{cl}$	15.5±.51	11.1±.58
full KYT, fixed $\bar{\sigma}_k = 1$	15.6±.49	11.3±.56
Per-class loss reweighting	15.6±.46	11.2±.57
Per-class learning rate scaling	15.4±.50	10.9±.61

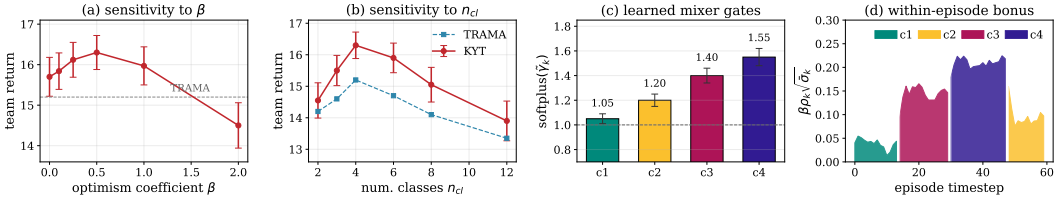


Figure 6: Sensitivity and diagnostics on SurComb4: (a) optimism coefficient β (dashed: TRAMA), (b) class count n_{cl}, (c) learned mixer gates, and (d) within-episode exploration bonuses colored by predicted class.

$\mathrm{softplus}(\tilde{\gamma}_k)$. The values are not uniform across classes, with the harder classes receiving a larger multiplicative gate, consistent with class-dependent value scaling. Panel (d) shows the realized optimism bonus along a single evaluation episode, color-coded by the predicted class at each timestep. The bonus reacts to within-episode class transitions: it spikes when the agents enter a region the identification module flags as a hard class and decays as the team re-enters easy regions. This within-episode reactivity is what allows KYT to inherit class-conditioned behavior even though the class label is recomputed at every step.

5 Related Work

Class-aware MARL. Recent work recovers task or sub-task identity from interaction data and conditions the policy on the recovered label. RODE clusters per-agent action embeddings into roles [Wang et al., 2021b], LDSA performs latent sub-task assignment with a softmax over learned sub-task embeddings [Yuan et al., 2022], MASER selects sub-goals from the replay buffer [Jeon et al., 2022], and TRAMA [Na et al., 2025] clusters quantized trajectory embeddings with K-means and feeds the class representation to the policy. These methods develop representations of class, role, or sub-task structure. KYT complements this direction by using class identity to guide the Bellman update and mixer modulation.

Multi-task MARL and cross-task generalization. A separate line of multi-task MARL learns coordination skills that transfer across tasks with different numbers and types of agents, typically from offline data, and evaluates on unseen tasks [Zhang et al., 2023, Liu et al., 2025, Jeon et al., 2026]. Our setting is narrower: latent trajectory classes within one environment, a fixed agent set, and no held-out tasks. The two are in principle composable, since a cross-task method that produces a per-instance label could supply the label that KYT consumes.

Value factorization, exploration, and curriculum. Value factorization for cooperative MARL spans VDN [Sunehag et al., 2018], QMIX [Rashid et al., 2020b], QTRAN [Son et al., 2019], QPLEX [Wang et al., 2021a], Qatten [Yang et al., 2020], WQMIX [Rashid et al., 2020a], and FACMAC [Peng et al., 2021]; centralized-critic alternatives include COMA [Foerster et al., 2018] and MADDPG [Lowe et al., 2017], alongside fully decentralized learners such as MA2QL [Su et al., 2022] and sequence-modeling approaches such as MAT [Wen et al., 2022]. The class-conditioned modulation in Equation 7 adapts feature-wise linear modulation [Perez et al., 2018] to the cooperative

MARL mixer and is a parameter-light alternative to hypernetwork-generated mixer weights [Ha et al., 2017]. MARL exploration methods such as MAVEN [Mahajan et al., 2019], EMC [Zheng et al., 2021], CDS [Li et al., 2021], EITI/EDTI [Wang et al., 2020b], EOI [Jiang and Lu, 2021], and CMAE [Liu et al., 2021] offer complementary approaches to cooperative exploration. The bonus in Equation 5 is class-relative through ρ_k and concentrates exploration on classes that still show disagreement. A separate line of work reshapes the data distribution or the training signal according to sample or class difficulty, through curricula and prioritized replay [Bengio et al., 2009, Schaul et al., 2016, Narvekar et al., 2020, Li et al., 2026] or through class-level and dynamics-aware reweighting in supervised learning [Li et al., 2023, 2025]. KYT modifies the Bellman target using class-level disagreement among the agents.

Class-level learning dynamics. From a learning-dynamics perspective, trajectory classes provide a natural unit for aggregating training signals across episodes. This connects class-aware MARL with the broader idea of adapting training to differences in difficulty [Narvekar et al., 2020, Li et al., 2023, 2025]. In KYT, the evolving class statistics guide the allocation of optimistic bonuses, while the class-conditioned mixer supplies a corresponding value transformation. This perspective connects trajectory representation, exploration, and value learning through their shared class structure.

6 Conclusion

We presented Know Your Task, Learn It Right (KYT), a class-aware extension of value factorization for cooperative MARL over latent trajectory classes that turns class identification from a policy input into a control signal for the Bellman update. KYT maintains a parameter-free per-class dynamics tracker, uses it to add a bounded, class-relative optimistic bonus to the TD target, and applies a softplus-gated affine modulation to the mixer to make credit assignment class-conditional. Both mechanisms preserve the IGM property of the underlying factorization. Across SMACv2, SurComb, and SMAC benchmarks the method outperforms strong class-aware and exploration-aware baselines, with the largest gains on the hardest classes.

Limitations. KYT allocates exploration pressure across classes, not across states within a class. Because the bonus is driven by cross-agent disagreement, a class on which agents agree confidently on low values, for example a passive local optimum in which allied units learn to retreat or stall, receives no additional pressure, and within-class exploration remains that of the base method (ϵ-greedy here). The bonus enters the target additively, so a state-level novelty term composes with it directly, and the policy entropy $\bar{H}_k$ already recorded by the tracker distinguishes a confidently stuck class (low disagreement, low entropy, low return) from a solved one; using $(\bar{\sigma}_k, \bar{H}_k)$ jointly is a concrete next step. Second, disagreement is not identical to remaining learnability: agents with different roles retain a structural floor of $\bar{\sigma}_k$, so ρ_k can keep mass on a class with little left to learn. Subtracting an estimated floor would restore exact asymptotic recovery, but we have not evaluated this variant, and we have not compared $\bar{\sigma}_k$ against an independent uncertainty estimator such as an ensemble or random network distillation [Burda et al., 2019]. Third, with near-homogeneous agents $\bar{\sigma}_k$ is small everywhere and KYT approaches its base method. Fourth, ρ_k assumes that every class is visited often enough for its EMA to remain comparable; domains with very rare classes may need a stratified sampler. Fifth, n_{cl} must be fixed in advance. Over-partitioning degrades gracefully because ρ_k splits mass among sub-classes of similar difficulty, whereas under-partitioning merges classes with different dynamics and reinstates the averaging that KYT removes (Figure 6b). Finally, the disagreement signal is defined for cooperative teams; mixed cooperative-competitive settings will need a different aggregation rule, and extending class-aware Bellman adaptation to embodied planning, where task structure must be discovered without human prior knowledge [Liu, 2026], is a natural direction for future work.

Acknowledgments and Disclosure of Funding

The authors acknowledge the contributions of the research community in developing the methods and benchmark environments that support this work. This research received no external funding. The authors declare that they have no competing financial interests related to this work.

References

Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron Courville, and Marc Bellemare. Deep reinforcement learning at the edge of the statistical precipice. In Advances in Neural Information Processing Systems, volume 34, pages 29304–29320, 2021.

Adrià Puigdomènech Badia, Pablo Sprechmann, Alex Vitvitskyi, Daniel Guo, Bilal Piot, Steven Kapturowski, and Olivier Tieleman. Never give up: Learning directed exploration strategies. In International Conference on Learning Representations, 2020.

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In International Conference on Machine Learning, 2009.

Yuri Burda, Harrison Edwards, Amos Storkey, and Oleg Klimov. Exploration by random network distillation. In International Conference on Learning Representations, 2019.

Benjamin Ellis, Jonathan Cook, Skander Moalla, Mikayel Samvelyan, Mingfei Sun, Anuj Mahajan, Jakob N Foerster, and Shimon Whiteson. SMACv2: An improved benchmark for cooperative multi-agent reinforcement learning. In Advances in Neural Information Processing Systems, volume 36, 2023.

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.

David Ha, Andrew M. Dai, and Quoc V. Le. HyperNetworks. In International Conference on Learning Representations, 2017.

Jeewon Jeon, Woojun Kim, Whiyoung Jung, and Youngchul Sung. MASER: Multi-agent reinforcement learning with subgoals generated from experience replay buffer. In International Conference on Machine Learning, pages 10041–10052, 2022.

Jiwon Jeon, Myungsik Cho, and Youngchul Sung. STAIRS-Former: Spatio-temporal attention with interleaved recursive structure transformer for offline multi-task multi-agent reinforcement learning. arXiv preprint arXiv:2603.11691, 2026.

Jiechuan Jiang and Zongqing Lu. The emergence of individuality. In International Conference on Machine Learning, 2021.

Chenghao Li, Tonghan Wang, Chengjie Wu, Qianchuan Zhao, Jun Yang, and Chongjie Zhang. Celebrating diversity in shared multi-agent reinforcement learning. In Advances in Neural Information Processing Systems, volume 34, pages 3991–4002, 2021.

Mengyang Li, Fengguang Su, Ou Wu, and Ji Zhang. Class-level logit perturbation. IEEE Transactions on Neural Networks and Learning Systems, 2023.

Mengyang Li, Xiaoling Zhou, and Ou Wu. Delving into the training dynamics for image classification. IEEE Transactions on Image Processing, 2025.

Mengyang Li, Haozhan Geng, Zhong Zhang, and Shuang Liu. Dual-difficulty curriculum learning for direct preference optimization. In ACM SIGKDD Conference on Knowledge Discovery and Data Mining, 2026.

Boyin Liu, Zhiqiang Pu, Yi Pan, Jianqiang Yi, Yanyan Liang, and Du Zhang. Lazy agents: A new perspective on solving sparse reward problem in multi-agent reinforcement learning. In International Conference on Machine Learning, pages 21937–21950, 2023.

Chang Liu. Self-adaptive counterfactual open-loop planning without human prior knowledge. In IEEE/RSJ International Conference on Intelligent Robots and Systems, 2026.

Iou-Jen Liu, Unnat Jain, Raymond A Yeh, and Alexander Schwing. Cooperative exploration for multi-agent deep reinforcement learning. In International Conference on Machine Learning, 2021.

Sicong Liu et al. Learning generalizable skills from offline multi-task data for multi-agent cooperation. In International Conference on Learning Representations, 2025.

Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. In Advances in Neural Information Processing Systems, volume 30, 2017.

Anuj Mahajan, Tabish Rashid, Mikayel Samvelyan, and Shimon Whiteson. MAVEN: Multi-agent variational exploration. In Advances in Neural Information Processing Systems, volume 32, 2019.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, and Alex Graves. Human-level control through deep reinforcement learning. Nature, 518 (7540):529–533, 2015.

Hyungho Na, Kwanghyeon Lee, Sumin Lee, and Il-Chul Moon. Trajectory-class-aware multi-agent reinforcement learning. In International Conference on Learning Representations, 2025.

Sanmit Narvekar, Bei Peng, Matteo Leonetti, Jivko Sinapov, Matthew E. Taylor, and Peter Stone. Curriculum learning for reinforcement learning domains: A framework and survey. Journal of Machine Learning Research, 21(181):1–50, 2020.

Frans A. Oliehoek and Christopher Amato. A concise introduction to decentralized POMDPs, volume 1. Springer International Publishing, 2016.

Ian Osband, Charles Blundell, Alexander Pritzel, and Benjamin Van Roy. Deep exploration via bootstrapped DQN. In Advances in Neural Information Processing Systems, 2016.

Deepak Pathak, Pulkit Agrawal, Alexei A Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction. In International Conference on Machine Learning, pages 2778–2787, 2017.

Deepak Pathak, Dhiraj Gandhi, and Abhinav Gupta. Self-supervised exploration via disagreement. In International Conference on Machine Learning, pages 5062–5071, 2019.

Bei Peng, Tabish Rashid, Christian Schroeder de Witt, Pierre-Alexandre Kamienny, Philip Torr, Wendelin Böhmer, and Shimon Whiteson. FACMAC: Factored multi-agent centralised policy gradients. In Advances in Neural Information Processing Systems, volume 34, pages 12208–12221, 2021.

Ethan Perez, Florian Strub, Harm De Vries, Vincent Dumoulin, and Aaron Courville. FiLM: Visual reasoning with a general conditioning layer. In AAAI Conference on Artificial Intelligence, 2018.

Kate Rakelly, Aurick Zhou, Chelsea Finn, Sergey Levine, and Deirdre Quillen. Efficient off-policy meta-reinforcement learning via probabilistic context variables. In International Conference on Machine Learning, pages 5331–5340, 2019.

Tabish Rashid, Gregory Farquhar, Bei Peng, and Shimon Whiteson. Weighted QMIX: Expanding monotonic value function factorisation for deep multi-agent reinforcement learning. In Advances in Neural Information Processing Systems, volume 33, pages 10199–10210, 2020a.

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, 2020b.

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 International Conference on Autonomous Agents and Multi-Agent Systems, pages 2186–2188, 2019.

Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay. In International Conference on Learning Representations, 2016.

Kyunghwan Son, Daewoo Kim, Wan Ju Kang, David Earl Hostallero, and Yung Yi. QTRAN: Learning to factorize with transformation for cooperative multi-agent reinforcement learning. In International Conference on Machine Learning, pages 5887–5896, 2019.

Alexander L Strehl and Michael L Littman. An analysis of model-based interval estimation for Markov decision processes. Journal of Computer and System Sciences, 74(8):1309–1331, 2008.

Kefan Su, Siyuan Zhou, Jiechuan Jiang, Chuang Gan, Xiangjun Wang, and Zongqing Lu. Ma2ql: A minimalist approach to fully decentralized multi-agent reinforcement learning. arXiv preprint arXiv:2209.08244, 2022.

Peter Sunehag, Guy Lever, Audrunas Gruslys, Wojciech Marian Czarnecki, Vinicius Zambaldi, Max Jaderberg, and Marc Lanctot. Value-decomposition networks for cooperative multi-agent learning based on team reward. In International Conference on Autonomous Agents and Multi-Agent Systems, pages 2085–2087, 2018.

Jianhao Wang, Zhizhou Ren, Terry Liu, Yang Yu, and Chongjie Zhang. QPLEX: Duplex dueling multi-agent Q-learning. In International Conference on Learning Representations, 2021a.

Tonghan Wang, Heng Dong, Victor Lesser, and Chongjie Zhang. ROMA: Multi-agent reinforcement learning with emergent roles. In International Conference on Machine Learning, 2020a.

Tonghan Wang, Jianhao Wang, Yi Wu, and Chongjie Zhang. Influence-based multi-agent exploration. In International Conference on Learning Representations, 2020b.

Tonghan Wang, Tarun Gupta, Anuj Mahajan, Bei Peng, Shimon Whiteson, and Chongjie Zhang. RODE: Learning roles to decompose multi-agent tasks. In International Conference on Learning Representations, 2021b.

Muning Wen, Jakub Kuba, Runji Lin, Weinan Zhang, Ying Wen, Jun Wang, and Yaodong Yang. Multi-agent reinforcement learning is a sequence modeling problem. In Advances in Neural Information Processing Systems, volume 35, pages 16509–16521, 2022.

Yaodong Yang, Jianye Hao, Ben Liao, Kun Shao, Guangyong Chen, Wulong Liu, and Hongyao Tang. Qatten: A general framework for cooperative multi-agent reinforcement learning. arXiv preprint arXiv:2002.03939, 2020.

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. In Advances in Neural Information Processing Systems, volume 35, pages 24611–24624, 2022.

Lei Yuan, Chenghe Wang, Jianhao Wang, Fuxiang Zhang, Feng Chen, Cong Guan, Zongzhang Zhang, Chongjie Zhang, and Yang Yu. Multi-agent concentrative coordination with decentralized task representation. In International Joint Conference on Artificial Intelligence, pages 599–605, 2022.

Fuxiang Zhang et al. Discovering generalizable multi-agent coordination skills from multi-task offline data. In International Conference on Learning Representations, 2023.

Lulu Zheng, Jiarui Chen, Jianhao Wang, Jiamin He, Yujing Hu, Yingfeng Chen, Changjie Fan, Yang Gao, and Chongjie Zhang. Episodic multi-agent reinforcement learning with curiosity-driven exploration. In Advances in Neural Information Processing Systems, volume 34, pages 3757–3769, 2021.

A Extended Theoretical Analysis

In this section we give the proof of Proposition 1, report the residual disagreement floor that motivates its general form, and analyze the noise of the EMA-based tracker.

A.1 Proof of Proposition 1

Part 1: Boundedness of the exploration bonus. Recall the definition of the cross-agent value disagreement EMA for class k:

$$\bar{\sigma}_k(T) = (1-\eta)\,\bar{\sigma}_k(T-1) + \eta \cdot \mathrm{Var}_i\big[Q^i_\theta(\tau^i, a^i)\big]\,\big|_k. \tag{9}$$

Since the variance $\mathrm{Var}_i[Q^i]$ is non-negative by definition, and $\eta \in (0,1)$, a simple inductive argument ensures that $\bar{\sigma}_k(T) \geq 0$ for all classes $k \in \{1,\ldots,n_{cl}\}$ and all training steps $T \geq 0$, assuming $\bar{\sigma}_k(0) \geq 0$.

The relative weight is defined as:

$$\rho_k(T) = \frac{\bar{\sigma}_k(T)}{\sum_{k'=1}^{n_{cl}} \bar{\sigma}_{k'}(T) + \epsilon}. \tag{10}$$

Given $\epsilon > 0$ and $\bar{\sigma}_{k'} \geq 0$, it strictly holds that $0 \leq \rho_k(T) < 1$. The exploration bonus added inside the bracket of Equation 5 is $B_{\hat{k}}(T) = \beta\,\rho_{\hat{k}}(T)\,\sqrt{\bar{\sigma}_{\hat{k}}(T)}$. Let $\bar{\sigma}_{\max}(T) = \max_k \bar{\sigma}_k(T)$. Since $\rho_{\hat{k}}(T) \leq 1$ and $\bar{\sigma}_{\hat{k}}(T) \leq \bar{\sigma}_{\max}(T)$, we obtain

$$0 \leq B_{\hat{k}}(T) \leq \beta\,\sqrt{\bar{\sigma}_{\max}(T)}. \tag{11}$$

The bonus is therefore bounded at every step by the largest current class-level disagreement, independently of η.

Part 2: Limiting behavior. Suppose $\bar{\sigma}_k(T) \to \bar{\sigma}^\infty_k \geq 0$ for every class k. The map $(\bar{\sigma}_1,\ldots,\bar{\sigma}_{n_{cl}}) \mapsto \rho_k$ is continuous on the non-negative orthant because its denominator is at least $\epsilon > 0$, and $x \mapsto \sqrt{x}$ is continuous on $[0,\infty)$. Hence $\rho_k(T) \to \rho^\infty_k = \bar{\sigma}^\infty_k / (\sum_{k'} \bar{\sigma}^\infty_{k'} + \epsilon)$ and

$$B_k(T) \to c_k := \beta\,\rho^\infty_k\,\sqrt{\bar{\sigma}^\infty_k} \;\leq\; \beta\,\sqrt{\bar{\sigma}^\infty_{\max}}, \tag{12}$$

where the inequality follows from Part 1 in the limit. The target of Equation 5 thus converges to $y_{\hat{k}} \to r + \gamma\big[\max_{\mathbf{a}'} Q^{\theta^-}_{tot}(\mathbf{s}', \mathbf{a}') + c_{\hat{k}}\big]$. If $\bar{\sigma}^\infty_k = 0$ for every class, then $c_k = 0$ and $y_{\hat{k}}$ converges to the standard one-step Bellman target.

To interpret a non-zero limit, consider the exact (tabular) optimality operator with the limiting bonus, $(\mathcal{T}_c Q)(s,\mathbf{a}) = r + \gamma\,\mathbb{E}[\max_{\mathbf{a}'} Q(s',\mathbf{a}') + c_{k(s')}]$, and the standard operator $\mathcal{T}$ with fixed point Q^*. If all c_k equal a common constant c, substituting $Q = Q^* + d$ into $Q = \mathcal{T}_c Q$ gives $d = \gamma d + \gamma c$, so the fixed point of $\mathcal{T}_c$ is $Q^* + \gamma c/(1-\gamma)$: a state-independent offset that leaves the greedy policy unchanged. If the c_k differ, $\|\mathcal{T}_c Q - \mathcal{T} Q\|_\infty \leq \gamma \max_k c_k$ for every Q, and since both operators are γ-contractions their fixed points differ by at most $\gamma \max_k c_k/(1-\gamma)$ in the sup-norm. In finite-horizon episodic tasks the argument for equal c_k no longer gives an exactly policy-neutral offset, because states closer to termination accumulate fewer bonus terms; this induces a mild preference for longer episodes, whose size is bounded by the per-step residual reported in Appendix A.2.

Part 3: IGM preservation. The Individual-Global-Max (IGM) condition is defined as:

$$\arg\max_{\mathbf{a}} Q_{tot}(\mathbf{s},\mathbf{a}) = \left(\arg\max_{a^1} Q^1(\tau^1,a^1),\ldots,\arg\max_{a^n} Q^n(\tau^n,a^n)\right). \tag{13}$$

KYT modulates the base factorization via $Q_{tot}(\mathbf{s},\mathbf{a};\hat{k}) = \mathrm{softplus}(\tilde{\gamma}_{\hat{k}}) \cdot Q^{base}_{tot}(\mathbf{s},\mathbf{a}) + \beta^{mix}_{\hat{k}}$. Let $f(x) = c_1 x + c_2$, where $c_1 = \mathrm{softplus}(\tilde{\gamma}_{\hat{k}})$ and $c_2 = \beta^{mix}_{\hat{k}}$. Since $\mathrm{softplus}(z) = \log(1+e^z) > 0$ for all $z \in \mathbb{R}$, we have $c_1 > 0$. Thus, $f(x)$ is a strictly monotonically increasing linear function. For any strictly monotonically increasing function f, $\arg\max_x f(g(x)) = \arg\max_x g(x)$. Therefore:

$$\arg\max_{\mathbf{a}} Q_{tot}(\mathbf{s},\mathbf{a};\hat{k}) = \arg\max_{\mathbf{a}} Q^{base}_{tot}(\mathbf{s},\mathbf{a}). \tag{14}$$

Since Q^{base}_{tot} (e.g., QPLEX or QMIX) already satisfies the IGM condition, the modulated Q_{tot} inherits it. The argument holds for every value of $\hat{k}$, so it does not require the predicted label to be correct.

A.2 Residual disagreement floor

Part 2 of Proposition 1 is stated for a general limit because $\bar{\sigma}_k$ need not converge to zero. Table 4 reports the terminal $\bar{\sigma}$ under KYT at 5M steps on the four SMAC maps, where unit heterogeneity and difficulty can be separated. The residual tracks heterogeneity rather than difficulty: the easiest map (1c3s5z) retains a substantially higher floor than the hardest (6h_vs_8z). The floor therefore reflects structural role differences between agents, not leftover epistemic uncertainty.

Table 4: Terminal cross-agent disagreement under KYT at 5M steps versus unit composition and final win rate (Table 1).

Map	Unit composition	Win rate	$\bar{\sigma}$ at 5M
5m_vs_6m	homogeneous (Marines)	0.86	0.04
6h_vs_8z	homogeneous (Hydralisks)	0.79	0.09
1c3s5z	heterogeneous (3 unit types)	0.97	0.23
MMM2	heterogeneous (incl. healer)	0.92	0.31

To separate a stationary from a decaying component within a benchmark, we fit $\bar{\sigma}_k(T) = a_k + b_k \exp(-T/\tau_k)$ per class. On SurComb4 the fitted floors are $a_k = 0.048, 0.045, 0.056, 0.089$ for classes 1 to 4, with $R^2 = 0.990, 0.993, 0.995, 0.995$; the decaying term b_k vanishes in the limit while a_k does not. The first three agree on a common value near 0.05, so on this benchmark the floor is a property of the map rather than of the class; the fourth is higher only because its transient is slower ($\tau_4 \approx 1.4$M against 0.40–0.92M) and still contributes 0.04 at 5M, so the fitted a_4 is an upper bound. On MMM2, which has the highest floor, the per-step bound of Proposition 1 is $\beta\sqrt{\bar{\sigma}^{\infty}_{\max}} \approx 0.28$ ($\beta = 0.5$, $\bar{\sigma}^{\infty} = 0.31$), and the realized bonus, which carries the further factor ρ^{∞}_k, is about 0.09. On MMM2, KYT improves the win rate by five percentage points over TRAMA (0.92 vs. 0.87), while 1c3s5z, with the second-highest floor, saturates for both methods. A variant that replaces $\bar{\sigma}_k$ with $\max(0, \bar{\sigma}_k - \hat{a}_k)$, using a fitted or running-minimum estimate $\hat{a}_k$ of the floor, would restore exact asymptotic recovery and remove the episodic bias; we have not evaluated it.

A.3 Stability of the EMA-based tracker

We analyze the sampling variability of $\bar{\sigma}_k$ under EMA updates from replay-buffer mini-batches. Let $v_T = \mathrm{Var}_i[Q^i_\theta]|_k$ be the observed variance at step T. We model $v_T = \mu_k + \xi_T$, where μ_k is the expected disagreement for class k under the current policy, and ξ_T is sampling noise with $\mathbb{E}[\xi_T] = 0$ and $\mathrm{Var}(\xi_T) = \sigma^2_{noise}$, independent across steps. The EMA is given by $\bar{\sigma}_k(T) = (1-\eta)\bar{\sigma}_k(T-1) + \eta v_T$. Unrolling this yields:

$$\bar{\sigma}_k(T) = (1-\eta)^T \bar{\sigma}_k(0) + \eta \sum_{j=0}^{T-1} (1-\eta)^j (\mu_k + \xi_{T-j}). \tag{15}$$

Taking the expectation, $\mathbb{E}[\bar{\sigma}_k(T)] \to \mu_k$ as $T \to \infty$. The variance of the tracker is:

$$\mathrm{Var}(\bar{\sigma}_k(T)) = \eta^2 \sum_{j=0}^{T-1} (1-\eta)^{2j} \sigma^2_{noise} \approx \frac{\eta}{2-\eta} \sigma^2_{noise}. \tag{16}$$

With $\eta = 0.01$, the variance of the tracking signal is attenuated by a factor of roughly 0.005 relative to the raw batch variance under this model, so $\bar{\sigma}_k$ provides a smooth control signal for the Bellman target and suppresses high-frequency oscillations in the exploration bonus. The parameter η does not enter any statement of Proposition 1; it affects how quickly the tracker identifies hard classes, not the validity of the update.

B Experimental and Implementation Details

B.1 Benchmark environments

We provide detailed descriptions of the testbeds used in our experiments.

SMACv2: SMACv2 introduces stochasticity and dynamic spawn points to the original StarCraft Multi-Agent Challenge. Unlike SMAC, where agents can memorize deterministic sequences, SMACv2 requires generalization. We use `protoss5_vs_5` and `terran5_vs_5`. The variation across episodes arises from the randomized initial positions and enemy unit compositions.

SurComb & reSurComb: SurComb is a class-structured environment designed to test role discovery and class identification. It combines different unit types (e.g., Marines, Zealots, Stalkers) with sparse rewards. The classes ($n_{cl} = 4$ in all our runs) correspond to latent distributions of enemy combinations and maps. `reSurComb` variants are structurally identical but feature significantly sparser reward functions and randomized agent IDs, which makes cross-agent credit assignment harder. We report team return, i.e., the sum of the per-agent return over the five allied agents.

Class alignment across seeds. Codebook indices produced by the identification module are arbitrary and differ across seeds. For every per-class quantity we report (Tables 2 and 4, Figures 1, 3, 5, and 6c), classes are aligned across seeds by the enemy composition that defines them, not by codebook index, and ordered from easiest to hardest by TRAMA's final per-class return.

B.2 Network architectures

KYT is implemented in PyTorch with the shared backbone described in Section 4.1.

- **Agent Network:** A DRQN containing a 64-dimensional fully-connected layer, followed by a 64-dimensional GRU, and a final fully-connected layer outputting the action values.
- **Mixer Network (Base):** We use QPLEX as the default base mixer for all class-structured environments. The hypernetworks mapping states to mixer weights consist of 2-layer MLPs with a 64-dimensional hidden state and ReLU activations.
- **Trajectory Encoder (Identification Module):** Following Na et al. [2025], the historical trajectory is processed by a 1D-CNN with a kernel size of 3, followed by an LSTM layer. The output is quantized via a Vector Quantized (VQ) codebook with n_{cl} embeddings.
- **KYT Class-Conditioned Gating:** The gating network ω (Equation 7) is a lightweight MLP with one hidden layer. Input: one-hot encoded class label $\hat{k}$ (size n_{cl}). Hidden layer: 32 units with ReLU. Output: 2 units mapping to $\tilde{\gamma}_k$ and β_k^{mix}. For $n_{cl} = 4$ this adds $n_{cl} \times 32 + 32 \times 2 + 34 = 226$ parameters including biases, i.e., about 200.

B.3 Computational cost

Table 5 compares KYT with TRAMA. The tracker adds a variance reduction over the agent dimension of an already materialized tensor, three EMA updates, and one affine transform per step, with no extra forward or backward pass, buffer, or network. Wall-clock time is the mean of two runs; it is below the per-step overhead because environment interaction is identical for both methods. Execution cost is unchanged: the mixer and the tracker exist only in centralized training, and at execution each agent acts greedily on its own utility. All runs use NVIDIA RTX 4090 GPUs (24 GB), with 8 CPU cores per job and four jobs running concurrently per device. The runs reported here (ten tasks $\times$ eight methods $\times$ eight seeds for Table 1, i.e. 640 runs of 16.9 h plus the ablations, the identification-noise study and the hyperparameter sweeps) amount to about $15{,}000$ GPU-hours.

Table 5: Computational cost of KYT relative to TRAMA on SurComb4.

	TRAMA	KYT	Overhead
Learnable parameters	∼450k	∼450k + 226	$+0.05\%$
Milliseconds per gradient step	11.8	12.2	$+3.4\%$
Wall-clock time, 5M steps	16.4 h	16.9 h	$+3.0\%$
Peak GPU memory	3.12 GB	3.12 GB	unchanged

C Additional Empirical Evaluations

In this section, we present the data and analysis for the additional experiments referenced in the main text.

Table 6: Extended hyperparameter configuration.

Parameter	Value	Parameter	Value
Optimism coefficient β	0.5	Batch size (episodes)	32
EMA rate η	0.01	Replay buffer size	5000 episodes
Denominator floor ϵ	10^{-3}	Optimizer	Adam
Gating MLP hidden width	32	Learning rate	5×10^{-4}
Number of classes n_{cl}	4 (all suites, fixed)	Target update interval	200 steps
Discount factor γ	0.99	Exploration ϵ-greedy start	1.0
Codebook commitment loss (λ_{VQ})	0.25	Exploration ϵ-greedy end	0.05
Classification loss weight (λ_ζ)	0.1	Exploration annealing steps	50000
Total environment steps	5×10^6	Gradient clip norm	10.0

C.1 Final performance beyond the last evaluation point

Table 1 reports the last evaluation point of the 5M-step budget. To summarize performance over the final training window, Table 7 averages over the final 500k steps and reports the probability of improvement (PoI) of KYT over TRAMA [Agarwal et al., 2021], defined as the fraction of the 64 KYT/TRAMA seed pairs in which KYT attains the higher final return. On SurComb4, reSurComb3, and reSurComb4 the independent CIs do not overlap; on MMM2 and 6h_vs_8z they touch or overlap, and we rely on the PoI there.

Table 7: Final-window averages (last 500k steps, mean $\pm$ 95% CI over 8 seeds; computed for the two SurComb benchmarks) and probability of improvement of KYT over TRAMA at the final evaluation point.

Benchmark	TRAMA (final 500k)	KYT (final 500k)	PoI(KYT $>$ TRAMA)
SurComb4	15.08 ± 0.45	16.09 ± 0.39	0.94
reSurComb4	10.60 ± 0.57	12.01 ± 0.52	0.97
MMM2	–	–	0.91
6h_vs_8z	–	–	0.88

C.2 Validation of the disagreement signal

We assess the class ordering induced by $\bar{\sigma}_k$ using reward-based performance, cross-seed consistency, and environment-side structure. These checks complement its correlation with TD-error.

- **Reward-based performance.** Per-class final return is computed from environment reward. Within each benchmark, Spearman's rank correlation between $\bar{\sigma}_k$ and per-class final return is -1.00 on SurComb4 ($n = 4$), -1.00 on reSurComb4 ($n = 4$), -1.00 on SurComb3 ($n = 3$), and -0.50 on reSurComb3 ($n = 3$). Each n is small, so we report all four rather than a pooled figure.
- **Cross-seed stability.** We compare class orderings across independent initializations. On SurComb4, 6 of 8 seeds give the identical ordering, with a mean pairwise Spearman correlation of 0.91.
- **Environment-side structure.** SurComb classes are defined by enemy unit combinations, so a difficulty proxy is available from the environment specification alone. Using the enemy-to-ally ratio of total hit points plus shields at episode start, averaged per class, the $\bar{\sigma}_k$ ordering differs from the proxy by one adjacent transposition (Spearman 0.80, SurComb4).

Within a benchmark the allied composition is fixed and classes differ only in enemy composition, so the structural part of $\bar{\sigma}_k$ is approximately constant across classes and the differences between classes reflect the remaining learnable structure. Across maps the allied composition itself changes and the structural part dominates, which is what Table 4 measures. The uniform-ρ_k ablation in Table 3 gives functional evidence for the same conclusion. These checks support using $\bar{\sigma}_k$ to rank classes by remaining learning progress within a benchmark; the validated quantity is a relative difficulty indicator.

Table 8: Comparison of Bellman-target adaptation (KYT) with data-distribution and loss reweighting baselines. Final mean team returns $\pm$ 95% CI over 8 seeds.

Method Type	**Specific Variant**	**SurComb4**	**reSurComb4**
Uniform baseline	Base learner	15.20 ± 0.47	10.70 ± 0.59
Data/Loss Modification	PER-TD	15.35 ± 0.52	10.85 ± 0.63
	Per-class loss reweighting by ρ_k	15.60 ± 0.46	11.20 ± 0.57
Bellman Adaptation	**KYT (Ours)**	$\mathbf{16.30 \pm 0.42}$	$\mathbf{12.30 \pm 0.54}$

C.3 Comparison with data-level baselines

We compare Bellman-target adaptation with two data-level controls using the shared base learner: (1) **Prioritized Experience Replay (PER-TD)** [Schaul et al., 2016]: transitions are sampled from the replay buffer with probability proportional to their absolute TD-error; (2) **Per-Class Loss Reweighting**: transitions are sampled uniformly, but the TD loss is multiplied by the class weight $\rho_{\hat{k}}$.

As shown in Table 8, PER-TD provides small gains in this setting. In sparse-reward class-structured MARL, TD-error can reflect both learning progress and transition noise, which may affect the benefit of prioritization. Loss reweighting by ρ_k performs better, which is consistent with ρ_k being informative about class difficulty, while KYT attains the highest mean return. Scaling the gradient magnitude of a class does not change the target it regresses to, whereas the optimistic target in KYT changes the target on classes with high value disagreement.

C.4 Robustness to identification noise

KYT relies on the class $\hat{k}$ predicted by the identification module. Since the identification module is trained simultaneously, its early predictions can be noisy. To evaluate how sensitive KYT is to incorrect class assignments, we introduce a controlled ablation where a fraction $\xi \in [0, 0.5]$ of the predicted labels $\hat{k}$ are randomly scrambled during the Bellman update and tracker computation.

Table 9: Robustness of KYT on SurComb4 under varying levels of artificial label noise ξ. Final team return, mean $\pm$ 95% CI over 8 seeds.

Algorithm & Noise Level (ξ)	**Final Return**	**Change vs. $\xi = 0$**
Base learner	15.20 ± 0.47	–
KYT ($\xi = 0\%$, Default)	$\mathbf{16.30 \pm 0.42}$	0.0%
KYT ($\xi = 10\%$)	16.15 ± 0.44	-0.9%
KYT ($\xi = 20\%$)	15.80 ± 0.46	-3.1%
KYT ($\xi = 30\%$)	15.45 ± 0.49	-5.2%
KYT ($\xi = 50\%$)	15.05 ± 0.53	-7.7%

Table 9 shows that KYT degrades gracefully. With 10% and 20% scrambled labels KYT remains above the base learner; at 20% the confidence intervals overlap. At 50% noise, where the label carries little information, KYT falls to the baseline level. Two properties are consistent with this behavior: (1) the softplus gating in the mixer preserves IGM regardless of whether the label is correct (Proposition 1), so policy extraction stays consistent; (2) the EMA tracker with $\eta = 0.01$ acts as a low-pass filter that smooths out transient misclassifications.

C.5 Stratified vs. uniform sampling for the dynamics tracker

In Section 3.2, we chose to update the per-class tracker ψ_k online via an EMA using uniformly sampled transitions from the standard global replay buffer. An alternative design computes exact mini-batch averages via stratified sampling, i.e., forcing an equal number of transitions from each class in every training batch.

To test this, we implemented a variant called **KYT-BatchAvg**. This variant maintains separate replay buffers for each of the n_{cl} classes. During training, it samples equally from all buffers and

Table 10: Replay buffer strategies for dynamics tracking on SurComb4 ($n_{cl} = 4$). Standard KYT (uniform sampling + EMA) matches exact stratified sampling without per-class buffers.

Sampling Strategy	**Buffer Count**	**Est. RAM Usage**	**Tracking Update**	**Final Return**
Base learner	1 (Global)	~ 2.4 GB	N/A	15.20 ± 0.47
KYT-BatchAvg (Stratified)	4 (Per-class)	~ 9.6 GB	Exact Batch Variance	16.25 ± 0.45
KYT (Ours)	**1 (Global)**	$\sim \mathbf{2.4}$ **GB**	**EMA** ($\eta = 0.01$)	$\mathbf{16.30 \pm 0.42}$

updates the exploration bonus directly based on the exact batch variance, bypassing the EMA. The comparison is summarized in Table 10.

KYT-BatchAvg achieves a final return of 16.25 ± 0.45 on SurComb4, which is statistically indistinguishable from standard KYT (16.30 ± 0.42). However, KYT-BatchAvg requires n_{cl} separate replay buffers and routing logic, which increases buffer memory by a factor of n_{cl} (from 2.4 GB to nearly 10 GB for SurComb4). This supports our design choice: the uniform-sampling EMA acts as a low-pass filter (Appendix A.3) that approximates the per-class statistics without extra memory, provided the global buffer is large enough to contain recent samples from all classes. The floor $\epsilon = 10^{-3}$ in Equation 4 prevents division by zero when a class has not yet been visited.

C.6 Sensitivity to the EMA rate

The EMA rate η sets the time-scale of the tracker relative to the regression target. It should be small enough that $\bar{\sigma}_k$ varies more slowly than Q changes under gradient updates, which keeps the feedback loop between the tracker and target of Section 3.6 stable, and large enough that the tracker does not lag genuine policy improvement and keep bonus mass on mastered classes. Table 11 sweeps η on SurComb4. The non-default settings are 3-seed runs, for which we report mean $\pm$ half-range since 3 seeds are too few for a bootstrap CI; the default column is the 8-seed main result with its 95% CI. The three settings are within noise of one another, consistent with Appendix A.3. We used $\eta = 0.01$ on all benchmarks without per-environment tuning.

Table 11: Sensitivity to the EMA rate η on SurComb4 (final team return).

η	0.003 (3 seeds)	0.01 (default, 8 seeds)	0.03 (3 seeds)
Final return	16.18 ± 0.74	16.30 ± 0.42	16.07 ± 0.71

C.7 LAGMA on a subset of benchmarks

LAGMA [Liu et al., 2023] was run only on the six benchmarks shown in Figure 4 and is therefore not part of Table 1. Table 12 reports it on those six benchmarks. The relative performance across these benchmarks is consistent with the main results.

Table 12: LAGMA compared with TRAMA and KYT on the six benchmarks of Figure 4. Mean $\pm$ 95% CI over 8 seeds; the TRAMA and KYT columns repeat Table 1.

Benchmark	LAGMA	TRAMA	KYT
p5_vs_5	0.67±0.04	0.73±0.03	**0.77±0.03**
t5_vs_5	0.60±0.05	0.66±0.04	**0.70±0.03**
SurComb4	12.85±0.55	15.20±0.47	**16.30±0.42**
reSurComb4	8.35±0.66	10.70±0.59	**12.30±0.54**
MMM2	0.84±0.04	0.87±0.03	**0.92±0.02**
6h_vs_8z	0.67±0.05	0.73±0.04	**0.79±0.03**

C.8 Area under the curve

While Table 1 reports final performance, sample efficiency is also important in MARL. We therefore compute the area under the curve (AUC) of the learning trajectories across the 5 million training

Table 13: Area under the learning curve over 5×10^6 steps, obtained by integrating the curves of Figure 4 in closed form, $\int_0^T f\,(1 - e^{-rt})/(1 - e^{-rT})\,dt$ with the plotted final value f and rate r, so that every entry can be read off the corresponding panel. Higher AUC indicates faster learning and higher overall sample efficiency.

Task	**QPLEX**	**EMC**	**TRAMA**	**KYT (Ours)**
p5_vs_5	2.18×10^6	2.44×10^6	2.85×10^6	$\mathbf{3.17 \times 10^6}$
SurComb4	42.4×10^6	50.0×10^6	59.2×10^6	$\mathbf{67.0 \times 10^6}$
reSurComb4	25.7×10^6	32.5×10^6	41.1×10^6	$\mathbf{50.4 \times 10^6}$
MMM2	2.79×10^6	3.20×10^6	3.45×10^6	$\mathbf{3.81 \times 10^6}$

steps, which aggregates both the speed of learning and the final performance. Table 13 shows the largest AUC improvement on the hardest benchmark, `reSurComb4` (50.4×10^6 vs. TRAMA's 41.1×10^6, a 23% relative increase). The AUC gap exceeds the 15% gap in final return on the same map because KYT also reaches its plateau earlier, supporting an improvement throughout training. This is consistent with the Bellman dynamics in Figure 5: the class weight ρ_k moves bonus mass away from easy classes and toward hard classes early in training. Methods that spread exploration uniformly across classes, such as EMC, show correspondingly lower AUC.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction describe class-dependent learning dynamics, the KYT method, and the empirical and theoretical contributions detailed in Sections 3 and 4.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Yes, limitations are explicitly discussed in the Conclusion (Section 6), specifically addressing the assumption of class visitation frequency for the EMA tracker and the challenge of extending the method to mixed cooperative-competitive settings.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: Proposition 1 states the theoretical properties in Section 3.6. Appendix A provides the proofs and the EMA variance analysis with its assumptions.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: Section 4.1 and Appendix B describe hyperparameters, network architectures, baseline configurations, and evaluation metrics. The algorithm and training objective are specified in Section 3.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [No]

   Justification: The source code is not yet publicly available. We plan to release the implementation with installation and reproduction instructions; code release is coming soon. The paper provides the method and experimental configuration.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Training details, including the optimizer, learning rate, batch size, and model parameters, are provided in Section 4.1 and Table 6 in Appendix B.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: The main performance comparisons report means and 95% percentile bootstrap confidence intervals over eight independent seeds. Table 11 identifies the three-seed sensitivity runs and reports their half-ranges. Supplementary diagnostics and computational measurements state their respective reporting conventions.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: The compute workers (NVIDIA RTX 4090, 24 GB, eight CPU cores per job, four jobs per device), the per-run wall-clock time (16.9 h for 5M steps), and the reported project compute (about 15,000 GPU-hours) are described in Appendix B.3 and Table 5. Table 10 compares buffer memory usage.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: This research uses standard open-source reinforcement learning simulators and does not violate any aspect of the NeurIPS Code of Ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [N/A].

    Justification: This is foundational theoretical and algorithmic research on value factorization in multi-agent reinforcement learning applied to abstract video game simulators, which does not have immediate foreseeable negative societal impacts or direct applications to real-world deployment.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A].

    Justification: The planned release concerns an algorithm implementation for video game benchmarks and does not involve high-risk datasets or general-purpose pretrained models.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: The paper properly cites the creators of the SMAC, SMACv2, and SurComb environments, and our code respects their respective open-source licenses (e.g., MIT and Apache 2.0).

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [No]

Justification: The planned code release is not yet available. Installation and reproduction instructions will accompany the implementation. The paper currently documents the algorithm and experimental configuration.

Guidelines:

- The answer [N/A] means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [N/A].

Justification: This research involves simulated agents and does not involve crowdsourcing or human subjects.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A].

    Justification: This research does not involve human subjects, hence IRB approval is not applicable.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A].

    Justification: The core methodology of this research relies strictly on multi-agent reinforcement learning architectures and Bellman dynamics, and does not involve the use of Large Language Models as a core component.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.