BLUR: BOOTSTRAPPED LATENT UTILITY REWARD LEARNING FROM TRAJECTORY-LEVEL PERFORMANCE

Anonymous authors
Paper under double-blind review

ABSTRACT

Handcrafted dense rewards can substantially improve policy learning, but they require designers not only to assign numerical feedback, but also to specify which local behaviors represent task utility. Such specifications are necessarily selective and may omit useful behavioral regularities that are difficult to encode a priori. We propose **Bootstrapped Latent Utility Reward Learning (BLUR)**, which uses coarse trajectory-level performance to learn context-dependent associations between recurring behavioral patterns and task utility. BLUR employs policy-relative supervision to bootstrap learning under failure-dominated experience, then discards the temporary relative model and trains a new reward model with baseline-independent semantics for final policy optimization. An LSTM–VaDE reward model captures interaction history and latent behavioral structure, while trajectory-performance statistics over these patterns define dense shaping signals. Experiments on SMAC and HighwayEnv with off-policy and on-policy backbones show that BLUR can provide a competitive learned alternative to handcrafted dense rewards across the tested settings, with ablations supporting the staged training design and latent utility model. The learned utilities represent statistical associations rather than causal or return-preserving decompositions.

1 INTRODUCTION

Dense step-wise rewards can substantially improve exploration and policy learning, but their limitations extend beyond the cost of manual design. Constructing them requires task-specific expertise and often repeated redesign across environments (Ng et al., 1999; Gupta et al., 2022; Dann et al., 2023). More fundamentally, a handcrafted shaping function provides a designer-specified and necessarily selective encoding of task utility through chosen states, events, or intermediate objectives. Such specifications may emphasize utility structures anticipated by the designer while overlooking behavioral patterns that are informative for task progress but difficult to identify or express a priori.

Adaptive reward-learning methods reduce manual transition-level reward specification, but do not in general remove this underlying utility-specification problem. Learned rewards remain conditioned on the supervision exposed to the learner, such as demonstrations, preferences, success/failure statistics, or episodic outcomes (Ng et al., 2000; Christiano et al., 2017; Ma et al., 2025; Arjona-Medina et al., 2019), and therefore on which aspects of task utility that supervision reveals. Methods relying mainly on local statistics or successful experience may further struggle with context-dependent behaviors and failure-dominated early trajectories (Ma et al., 2025; Minoshima & Arai, 2026). This raises a broader question: *can task-relevant behavior–utility structure itself be learned from coarse trajectory-level evaluation rather than manually specified through local reward rules?*

We propose **Bootstrapped Latent Utility Reward Learning (BLUR)**. Our central insight is that handcrafted dense rewards encode task-utility structure, not merely numerical feedback. In StarCraft, for example, rewards associated with enemy eliminations, damage dealt, allied casualties, and final outcomes encode expert knowledge about which interaction patterns indicate task progress (Samvelyan et al., 2019). Rather than manually specifying such local utility relations, BLUR treats trajectory-level performance as weak supervision for learning associations between recurring contextual behavioral patterns and overall task quality.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

This shifts human prior knowledge from prescribing which transitions should receive reward to defining what constitutes a high-quality trajectory. Across trajectories, recurring behavioral patterns can exhibit systematic associations with higher or lower performance, which BLUR converts into dense shaping signals. The resulting structure is statistical rather than causal: BLUR does not assume a unique ground-truth utility decomposition, preserve trajectory return, or estimate causal marginal contributions.

Learning this structure from trajectory-level supervision introduces three challenges. **First, contextual ambiguity:** locally similar behaviors may have different utility associations under different interaction histories. **Second, cold-start discrimination:** when early trajectories are predominantly unsuccessful, baseline-independent performance labels may provide weak contrast among behaviors. **Third, semantic nonstationarity:** policy-relative evaluation can distinguish better from worse early trajectories, but its meaning changes as the policy improves and is therefore unsuitable as a fixed final reward semantics.

BLUR addresses these challenges with three corresponding designs. An LSTM (Graves, 2012) encoder and VaDE (Jiang et al., 2016) organize history-dependent behaviors into latent contextual patterns whose trajectory-performance statistics define dense shaping. A temporary Relative Utility Model uses policy-relative trajectory labels to bootstrap exploration under failure-dominated experience. Because these semantics evolve with the policy, BLUR then discards the relative model and trains a newly initialized Absolute Reward Model from baseline-independent trajectory-performance labels; the accepted model is frozen and used to optimize an independently initialized target policy together with the sparse terminal task outcome.

Our contributions are fourfold:

- We formulate reward learning as learning a task-relevant behavior–utility structure from coarse trajectory-level performance, providing a learned alternative to manually specifying dense transition-level utility.
- We develop a context-dependent latent utility model that combines temporal history encoding with latent behavioral clustering to associate recurring behavioral patterns with trajectory performance.
- We introduce a relative-to-absolute training framework: policy-relative supervision bootstraps learning under failure-dominated experience, while a separately learned absolute reward model provides fixed semantics for final policy optimization.
- We evaluate BLUR across cooperative and single-agent tasks with both off-policy and on-policy backbones, together with training-stage and reward-model ablations.

2 Related Work

Handcrafted reward shaping. Dense shaping injects task knowledge through selected intermediate events or auxiliary objectives and can substantially improve exploration and policy learning (Ng et al., 1999; Jaderberg et al., 2016; Samvelyan et al., 2019; Gupta et al., 2022; Dann et al., 2023). However, such rewards also impose a designer-specified and selective representation of task utility: the designer determines which local events matter and how they are weighted, potentially leaving useful behavioral regularities unmodeled. BLUR instead retains coarse trajectory-level task evaluation and learns contextual behavior–utility associations from interaction data.

Reward and imitation learning. PGRD optimizes reward parameters through policy gradients (Sorg et al., 2010), while inverse and adversarial imitation learning infer rewards or occupancy objectives from expert demonstrations (Ng et al., 2000; Ziebart et al., 2008; Ho & Ermon, 2016; Fu et al., 2017). Preference-based methods learn from trajectory or segment comparisons (Christiano et al., 2017; Ibarz et al., 2018; Lee et al., 2021; Park et al., 2022; Liu et al., 2022; Kim et al., 2023; Hu et al., 2024; Liu et al., 2023), while T-REX and D-REX use ranked demonstrations (Brown et al., 2019; 2020). These approaches reduce direct transition-level reward engineering, but task utility remains specified through the supervision channel provided to the learner. BLUR instead uses continuous trajectory-performance supervision from self-generated experience, so the prior specifies overall trajectory quality rather than local utility relations. Related self-generated methods include self-imitation (Oh et al., 2018) and trajectory-based shaping (Ma et al., 2025; Minoshima & Arai,

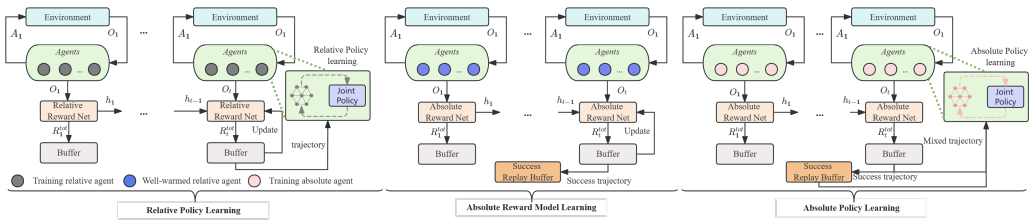

Figure 1: Conceptual BLUR training pipeline. The repeated panels depict successive interaction steps; the raster's A_1/O_1 labels correspond to generic A_t/O_t, and its observation label denotes contextual input $\mathbf{x}_t$. In the reported protocol, the accepted Absolute Reward Model is frozen and the final target policy is independently initialized. Domain-specific fitting and replay details are provided in Sections D–F.

2026). SASR estimates local utility from success/failure visitation statistics; BLUR instead models history-dependent behavioral patterns and separates temporary policy-relative bootstrap supervision from baseline-independent final shaping.

Episodic return redistribution and decomposition. RUDDER and Align-RUDDER redistribute delayed return using sequence modeling or aligned demonstrations (Arjona-Medina et al., 2019; Patil et al., 2020), while RRD, Diaster, and STAS learn trajectory-, sub-trajectory-, or spatial–temporal decompositions of episodic feedback (Ren et al., 2022; Lin et al., 2024; Chen et al., 2024). BLUR addresses a different objective: it learns contextual behavior–utility associations from trajectory performance without requiring return conservation or interpreting step-wise utility as causal temporal contribution. In cooperative MARL, BLUR supplies team-level shaping, leaving agent-wise credit assignment to the underlying optimizer (Foerster et al., 2018).

3 METHOD

We formulate BLUR under cooperative multi-agent reinforcement learning (MARL); the formulation reduces to the single-agent setting when only one agent is present. In the cooperative setting, BLUR supplies a centralized team-level shaping signal during training. It does not separately assign trajectory utility among agents and is not proposed as a solution to spatial agent-wise credit assignment. Agent policies continue to follow the decentralized execution interface of the underlying MARL optimizer. Background on VaDE and QMIX is provided in Sections B.1 and B.2, with implementation details in Section C.

3.1 OVERVIEW

Figure 1 illustrates BLUR's *Explore* $\rightarrow$ *Learn* $\rightarrow$ *Optimize* procedure. Rather than training the final reward model directly on early trajectories dominated by failures, BLUR separates policy-relative exploration supervision from the baseline-independent shaping semantics used for final policy optimization. The framework consists of *Relative Utility Bootstrap*, *Absolute Utility Learning*, and *Absolute Policy Optimization*.

Relative Utility Bootstrap (Explore). BLUR first initializes a Relative Utility Model and a relative policy. Each trajectory is evaluated relative to a moving baseline determined by recent policy performance. The resulting policy-referenced labels supervise a dense relative shaping signal used in an alternating loop of trajectory collection, reward-model learning, and policy optimization. This stage provides differentiated supervision even when successful trajectories are scarce.

Absolute Utility Learning (Learn). The temporary Relative Utility Model is subsequently discarded, and a new Absolute Reward Model is initialized from scratch. Retained and newly collected trajectories train the new model under baseline-independent trajectory-performance supervision. Here, "absolute" means independent of the moving policy baseline; it does not imply that the learned shaping reward preserves the numerical scale of P^{abs}. Sampling and readiness are implementation specific: SMAC uses stratified trajectory sampling and an explicit reward-quality gate,

whereas HighwayEnv uses general-trajectory sampling with successful-trajectory oversampling and a minimum-data/fit-round condition.

Absolute Policy Optimization (Optimize). Finally, BLUR freezes the accepted Absolute Reward Model, independently initializes the target policy, and optimizes that policy using dense shaping together with the sparse terminal task-success signal. Trajectory reuse follows the underlying optimizer: off-policy methods may reuse stored trajectories, whereas on-policy methods rely on fresh rollouts.

This staged design confines the policy-relative reward model to exploration and establishes absolute shaping with a separately initialized, baseline-independent model. It does not assert that every possible blended training scheme is inferior.

3.2 Problem Formulation

We consider a cooperative Dec-POMDP under centralized training and decentralized execution Bernstein et al. (2002); Oliehoek et al. (2016). A trajectory is denoted by $\tau = \{(\mathbf{o}_t, \mathbf{a}_t)\}_{t=1}^{T}$, where $\mathbf{o}_t$ and $\mathbf{a}_t$ are the joint observation and action. The reward model conditions on a contextual input $\mathbf{x}_t$, instantiated as the global state when centralized state information is available and otherwise as the environment observation. Agent policies continue to act only on local observations.

Rather than manually specifying transition-level utility, BLUR introduces a task-specific episode-level performance metric

$$P(\tau) \in \mathbb{R}. \tag{1}$$

This metric coarsely evaluates overall task quality through configured combinations of trajectory statistics, such as the final task outcome, cumulative damage, elimination ratio, or survival ratio. It provides no timestep supervision. Thus, BLUR retains task prior but shifts its design from local temporal utility rules to trajectory-level evaluation for learning context-dependent shaping associations.

Our objective is to learn a dense reward function

$$r_t^{\mathrm{BLUR}} = r_\theta(\mathbf{x}_t, \mathcal{H}_t). \tag{2}$$

where $\mathcal{H}_t$ denotes the trajectory history up to timestep t. Although the reward is indexed by the current encoding step t, $\mathbf{x}_t$ contains the consequence observed after the preceding joint action, and the resulting reward is assigned to the corresponding transition during policy optimization.

BLUR does not impose a return-conservation constraint requiring $\sum_t r_t^{\mathrm{BLUR}} = P(\tau)$, nor does it estimate the causal marginal contribution of each behavior. Instead, it treats trajectory-level performance as weak supervision for learning statistics associated with recurring contextual behavioral patterns. These statistics are converted into dense step-wise shaping signals for policy optimization.

3.3 Reward Model

The reward model estimates associations between contextual behavioral patterns and trajectory-level performance, then uses those associations to produce dense shaping signals. The complete architecture is illustrated in Section C.1 and consists of a Trajectory Behavior Encoder, a Latent Utility Clustering module, and a Utility Statistics module.

Trajectory Behavior Encoding. To incorporate interaction history, BLUR instantiates the temporal encoder with an LSTM Graves (2012) over the contextual sequence $\mathbf{x}_{1:T}$. Its hidden state is updated as

$$\mathbf{h}_t = f_{\mathrm{LSTM}}\left(\mathbf{x}_t, \mathbf{h}_{t-1}; \theta_{\mathrm{enc}}\right), \tag{3}$$

where $\mathbf{h}_t$ summarizes the interaction history up to time step t. This recurrent encoder is the reported implementation rather than the only possible temporal architecture; the evaluation includes an MLP encoder variant.

To make the hidden representation informative of task utility, an auxiliary performance-prediction head is introduced during training:

$$\widehat{P}(\tau) = f_{\text{perf}}\left(\text{Pool}\left(\mathbf{h}_1, \ldots, \mathbf{h}_T\right); \theta_{\text{perf}}\right). \tag{4}$$

The masked mean pooling operation and the treatment of variable-length trajectories are specified in Section C.2. Given the trajectory-level performance label $P(\tau)$ used in the current training stage, the encoder is optimized by

$$\mathcal{L}_{\text{enc}} = \frac{1}{|\mathcal{B}|} \sum_{\tau \in \mathcal{B}} \left(\widehat{P}(\tau) - P(\tau)\right)^2. \tag{5}$$

The prediction head is used only to supervise representation learning and is removed during reward inference.

Latent Utility Clustering. BLUR applies Variational Deep Embedding (VaDE) Jiang et al. (2016) to the encoded representations to identify recurring behavioral patterns. For each $\mathbf{h}_t$, VaDE produces a latent representation $\mathbf{z}_t$ and the posterior assignment probability

$$\gamma_{t,k} = p\left(c_t = k \mid \mathbf{z}_t\right), \qquad k \in \{1, \ldots, K\}, \tag{6}$$

where each latent cluster captures a group of behaviors with similar contextual characteristics.

VaDE does not directly predict rewards. Instead, it structures the representation space so that utility statistics can be estimated over recurring behavioral patterns. Its simplified optimization objective is

$$\mathcal{L}_{\text{VaDE}} = \lambda_{\text{rec}} \mathcal{L}_{\text{rec}} + \lambda_{\text{KL}} \mathcal{L}_{\text{KL}}, \tag{7}$$

where the reconstruction and mixture-regularization terms follow the adapted VaDE objectives implemented for the two domains. Reward inference uses deterministic cluster responsibilities evaluated at the posterior mean. Complete optimization details are provided in Section C.3.

In the reported implementation, the LSTM and VaDE are optimized sequentially rather than end-to-end. We first optimize the encoder using $\mathcal{L}_{\text{enc}}$, detach the resulting hidden representations, and then train VaDE independently; gradients from VaDE are not propagated to the LSTM encoder.

Utility Statistics and Reward Assignment. After clustering, BLUR associates each latent behavioral pattern with the performance labels of the trajectories in which it occurs. Given M encoded behavioral samples, let $\tau(m)$ denote the trajectory containing sample m, let $P(\tau(m))$ denote its trajectory-level performance label, and let $\gamma_{m,k}$ denote the posterior probability that behavioral sample m belongs to cluster k. The raw utility statistic of cluster k is computed as

$$\widetilde{U}_k = \frac{\sum_{m=1}^{M} \gamma_{m,k} P(\tau(m))}{\sum_{m=1}^{M} \gamma_{m,k} + \varepsilon}. \tag{8}$$

where ε is introduced for numerical stability. The utility U_k used for reward inference is obtained by normalizing the raw cluster statistics and clipping them to $[-1, 1]$; details are provided in Section C.4.

At time step t, the BLUR reward is obtained by posterior-weighted utility assignment:

$$r_t^{\text{BLUR}} = \sum_{k=1}^{K} \gamma_{t,k} U_k. \tag{9}$$

Here, U_k summarizes the trajectory-performance labels associated with latent behavioral pattern k, and r_t^{BLUR} is their posterior-weighted shaping assignment at timestep t. The estimator does not require the accumulated shaping reward to equal $P(\tau)$, and U_k is not a causal or marginal contribution of pattern k. Soft assignment avoids discontinuities near cluster boundaries and allows a behavior to exhibit characteristics of multiple latent patterns. Variable-length handling, normalization, and utility scaling are detailed in Sections C.2 and C.4.

The learned utility reward is combined with the terminal task-success signal according to the domain-specific policy-reward interface. The Relative Utility Model and the Absolute Reward Model share the same estimator family but differ in their trajectory-level labels, data, and code-level training schedules.

3.4 TRAINING DETAILS

The BLUR reward-learning pipeline is decoupled from the underlying policy optimizer. For concreteness, we present the main policy-update procedure using QMIX-style off-policy optimization. The same learned step-wise reward interface can also be used with on-policy actor–critic methods such as MAPPO, whose optimizer-specific data-management and update procedures are detailed in Sections D.4 and D.5.

Relative Utility Bootstrap. As illustrated in the left part of Figure 1, the relative policy collects trajectories evaluated by the trajectory-level function $P^{\mathrm{abs}}(\tau)$, which is not directly used as a step-wise reward. For trajectory τ_n, let b_n denote the mean absolute performance of a recent trajectory window. BLUR defines

$$P^{\mathrm{rel}}(\tau_n) = P^{\mathrm{abs}}(\tau_n) - b_n. \tag{10}$$

Because b_n depends on recent policy performance, P^{rel} is policy referenced and its semantics change as the policy changes. It distinguishes improvement over the recent policy level even before task success becomes frequent. The temporary Relative Utility Model converts these labels into posterior-weighted shaping values and is alternately updated with the relative policy. Once learned, its shaping value is

$$r_t^{\mathrm{rel}} = \sum_{k=1}^{K} \gamma_{t,k}^{\mathrm{rel}} U_k^{\mathrm{rel}}, \tag{11}$$

where $\gamma_{t,k}^{\mathrm{rel}}$ and U_k^{rel} are the relative cluster posterior and utility. The active bootstrap reward is implementation specific. SMAC begins with a raw trajectory-level fallback distributed across valid timesteps and, after its relative model passes a quality gate, uses a weighted mixture of raw-relative and learned-relative shaping. HighwayEnv instead fits the Relative Utility Model from an initial trajectory set before any policy update and combines its learned shaping with the terminal success signal. Each implementation stops the stage at its configured evaluation threshold or relative-policy update cap. The Relative Utility Model is then discarded so that its changing policy-relative semantics are not reused by the Absolute Reward Model.

Absolute Utility Learning. After bootstrapping, BLUR discards the Relative Utility Model and freezes the relative policy for absolute-model data collection. A newly initialized Absolute Reward Model is trained with baseline-independent labels $P^{\mathrm{abs}}(\tau)$. Data construction differs by domain. SMAC fits from a stratified sample of successful, high-performing failed, diverse failed, exploration, bootstrap, and recent absolute-policy trajectories, and repeats collection and fitting until a reward-quality gate is satisfied. HighwayEnv draws from the general trajectory buffer with successful-trajectory oversampling and transitions after its minimum trajectory and fit-round requirements are met. These procedures broaden behavioral coverage but do not guarantee complete support.

Using the shared reward-model architecture, BLUR estimates absolute cluster utilities and produces

$$r_t^{\mathrm{abs}} = \sum_{k=1}^{K} \gamma_{t,k}^{\mathrm{abs}} U_k^{\mathrm{abs}}, \tag{12}$$

Unlike the policy-relative label used during bootstrapping, $P^{\mathrm{abs}}(\tau)$ is baseline independent within a task. Once the Absolute Reward Model satisfies its domain-specific acceptance condition, its parameters are frozen for final policy training.

Absolute policy optimization. After Absolute Utility Learning, BLUR independently initializes the final target policy and trains it under the fixed absolute shaping semantics. This reset prevents the policy parameters learned under changing relative semantics from being carried into final optimization. HighwayEnv additionally initializes a fresh final replay buffer for DQN; other replay and rollout details remain optimizer specific.

As illustrated in the right part of Figure 1, the policy interacts with the environment under absolute shaping. The generic reward interface is

$$r_t^{\mathrm{final}} = \lambda_{\mathrm{abs}} r_t^{\mathrm{abs}} + \lambda_{\mathrm{succ}} r_t^{\mathrm{succ}}, \tag{13}$$

where r_t^{abs} is the dense shaping signal generated by the frozen Absolute Reward Model and r_t^{succ} is the terminal task-success or win/loss signal. The trajectory-level performance $P^{\mathrm{abs}}(\tau)$ supervises

reward-model learning but is not itself distributed as a step-wise policy reward. The domain-specific implementations set the coefficients and terminal outcome scale described in Section F. Throughout this stage, θ_{abs} remains fixed and only the target-policy parameters are updated.

Successful trajectories may be retained through replay suited to the optimizer. For off-policy methods, a whole update batch can be drawn from success-oriented replay; for on-policy methods, historical trajectories are not reused in the policy-gradient objective. Internal stage caps and accounting differ by domain, but reported learning curves and final-window results use common global-update support within each matched task–backbone comparison. Absolute-model fitting and success-replay collection perform no policy updates but add environment interactions, so the comparisons do not match total environment interaction.

4 EVALUATION

We evaluate BLUR through four research questions concerning policy performance, component behavior, optimizer and task compatibility, and qualitative representation structure.

RQ1: Can trajectory-performance-supervised utility learning provide a competitive alternative to handcrafted dense rewards?

RQ2: How do BLUR's staged training scheme and the examined reward-model components contribute to its effectiveness?

RQ3: Is BLUR compatible with representative off-policy and on-policy backbones, and can it be instantiated beyond cooperative MARL?

RQ4: Does BLUR exhibit qualitative utility-related structure in its learned representations?

Together, these questions assess the reported performance of trajectory-performance-supervised shaping, the examined training and model components, compatibility with representative policy backbones, and qualitative organization of the learned representations.

4.1 EXPERIMENTAL SETUP

Environments. We use SMAC Samvelyan et al. (2019) as our primary benchmark, with QMIX Rashid et al. (2018) and MAPPO Yu et al. (2022) as representative off-policy and on-policy backbones. To examine applicability beyond cooperative multi-agent control, we further instantiate BLUR with DQN Mnih et al. (2015) and PPO Schulman et al. (2017) on two HighwayEnv tasks Leurent et al. (2018). For HighwayEnv, test success is determined directly from the success information returned by the environment; no additional task-specific success criterion is introduced.

Baselines and ablations. We compare BLUR against the original handcrafted dense environment reward and SASR Ma et al. (2025), a representative self-generated, success/failure-conditioned reward-shaping method for sparse-reward learning. We independently reimplemented SASR following its published formulation and adapted it to the SMAC/QMIX interface; the HighwayEnv SASR results use PPO. Its learned shaping reward replaces the dense component while the sparse terminal outcome retains its native scale. SASR is not intended to represent the full episodic return-redistribution literature discussed in Section 2. We further evaluate two training-stage ablations and three reward-model ablations. Absolute Utility Learning performs no policy updates, although it may collect additional environment trajectories. Reported comparisons use common global-update support within each matched task–backbone setting; this does not imply matched total environment interaction. Domain-specific schedules are reported in Section H.6.

Evaluation protocol. We use test win rate on SMAC and test success rate on HighwayEnv. For each matched task–backbone comparison, evaluation records from all methods and seeds are linearly interpolated onto a common global policy-update grid with 500-update spacing over their shared support. The reporting horizon is the largest update jointly supported by every method and seed in that comparison. Final-window performance for each seed is the mean over the final $L = 10$ aligned grid points; we report the mean and sample standard deviation across $S = 5$ seeds. This matched global-update support avoids selecting a single noisy best checkpoint but does not imply matched total environment interaction. The complete statistical protocol and selected comparison-specific horizons are provided in Sections H.4 and H.5.

Table 1: Final-window test performance on SMAC and HighwayEnv (mean ± sample standard deviation over five seeds; each seed averages the final $L = 10$ aligned grid points). Off-policy/on-policy backbones are QMIX/MAPPO for SMAC and DQN/PPO for HighwayEnv; the single SASR HighwayEnv column uses PPO.

Task	Handcrafted Off-policy	Handcrafted On-policy	SASR	BLUR Off-policy	BLUR On-policy
SMAC					
1c3s5z	0.908 ± 0.064	0.930 ± 0.062	0.000 ± 0.000	0.839 ± 0.110	0.952 ± 0.024
2c_vs_64zg	0.000 ± 0.000	0.930 ± 0.023	0.162 ± 0.362	0.901 ± 0.030	0.953 ± 0.034
2s3z	0.890 ± 0.062	0.874 ± 0.073	0.000 ± 0.000	0.963 ± 0.025	0.902 ± 0.035
2s_vs_1sc	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.976 ± 0.020	0.968 ± 0.012
3m	0.878 ± 0.120	0.958 ± 0.081	0.000 ± 0.000	0.971 ± 0.036	0.972 ± 0.016
5m_vs_6m	0.000 ± 0.000	0.025 ± 0.013	0.000 ± 0.000	0.892 ± 0.045	0.023 ± 0.010
8m	0.938 ± 0.058	0.963 ± 0.044	0.000 ± 0.000	0.974 ± 0.027	0.961 ± 0.014
MMM2	0.944 ± 0.069	0.881 ± 0.080	0.000 ± 0.000	0.923 ± 0.067	0.972 ± 0.017
corridor	0.604 ± 0.078	0.000 ± 0.000	0.000 ± 0.000	0.882 ± 0.079	0.723 ± 0.066
HighwayEnv					
highway-v0	0.240 ± 0.303	0.234 ± 0.250	0.000 ± 0.000	0.960 ± 0.033	0.940 ± 0.013
roundabout-v0	0.704 ± 0.066	0.823 ± 0.055	0.000 ± 0.000	0.682 ± 0.076	0.882 ± 0.049

Table 2: Final-window QMIX ablation results on SMAC (mean ± sample standard deviation over five seeds; each seed averages the final $L = 10$ aligned grid points). Bold indicates the highest reported mean in each row and does not imply statistical significance.

	Complete	Training-stage Ablations		Reward-model Ablations		
Map	BLUR	Relative Only	Absolute Only	MLP+VaDE	BLUR-Direct	BLUR-KDE
1c3s5z	$\mathbf{0.839 \pm 0.110}$	0.681 ± 0.213	0.000 ± 0.000	0.763 ± 0.032	0.658 ± 0.240	0.723 ± 0.052
2c_vs_64zg	$\mathbf{0.901 \pm 0.030}$	0.517 ± 0.127	0.000 ± 0.000	0.822 ± 0.012	0.000 ± 0.000	0.124 ± 0.058
2s3z	$\mathbf{0.963 \pm 0.025}$	0.927 ± 0.011	0.835 ± 0.133	0.934 ± 0.036	0.568 ± 0.046	0.873 ± 0.014
2s_vs_1sc	$\mathbf{0.976 \pm 0.020}$	0.885 ± 0.087	0.000 ± 0.000	0.928 ± 0.018	0.936 ± 0.022	0.966 ± 0.016
3m	$\mathbf{0.971 \pm 0.036}$	0.901 ± 0.075	0.857 ± 0.136	0.965 ± 0.024	0.914 ± 0.048	0.908 ± 0.013
5m_vs_6m	$\mathbf{0.892 \pm 0.045}$	0.012 ± 0.002	0.000 ± 0.000	0.761 ± 0.087	0.000 ± 0.000	0.000 ± 0.000
8m	$\mathbf{0.974 \pm 0.027}$	0.881 ± 0.105	0.941 ± 0.028	0.946 ± 0.065	0.843 ± 0.085	0.897 ± 0.015
corridor	$\mathbf{0.882 \pm 0.079}$	0.000 ± 0.000	0.000 ± 0.000	0.630 ± 0.163	0.000 ± 0.000	0.000 ± 0.000
MMM2	$\mathbf{0.923 \pm 0.067}$	0.006 ± 0.001	0.000 ± 0.000	0.801 ± 0.085	0.000 ± 0.000	0.183 ± 0.054

4.2 Experimental Results

RQ1. We first evaluate this trajectory-performance-supervised alternative to handcrafted dense rewards. Table 1 summarizes late-training performance using the final-window protocol. BLUR improves over the corresponding handcrafted-reward baseline in most matched-backbone comparisons, while several off-policy and on-policy settings remain mixed. Large observed gains occur on challenging scenarios including 5m_vs_6m, corridor, and 2s_vs_1sc. For example, under the matched off-policy backbone, BLUR reports 0.892 ± 0.045 on 5m_vs_6m and 0.882 ± 0.079 on corridor, compared with 0.000 ± 0.000 and 0.604 ± 0.078 for the corresponding handcrafted-reward references.

Checkpoint-level learning dynamics are provided in Section I, whereas Table 1 reports final-window summaries. The displayed mean curves show earlier performance gains from handcrafted rewards in several scenarios, while BLUR maintains higher late-training test win rates on multiple challenging tasks. SASR remains at zero across all five seeds on several difficult settings. On 2c_vs_64zg, four of five SASR seeds remain at zero and only one seed exhibits non-zero performance, indicating substantial cross-seed instability under the evaluated setting. This pattern is consistent with cold-start sensitivity in success/failure-conditioned reward learning when successful trajectories are scarce. Because no significance tests are performed, small differences relative to the reported cross-seed variability should be interpreted cautiously, and SASR's zero-valued results alone do not establish general superiority over automatic reward-learning methods.

RQ2. We next examine the contributions of BLUR's staged training procedure and reward-model design. Table 2 reports both sets of ablations under the same five-seed final-window protocol.

Checkpoint-level dynamics of the stage variants are provided in Section I. *MLP+VaDE* replaces the recurrent encoder with an MLP while retaining VaDE; *BLUR-Direct* retains the LSTM but replaces VaDE utility estimation with direct utility regression; and *BLUR-KDE* retains the LSTM but uses Gaussian-kernel KDE utility estimation.

For the training-stage ablations, the complete BLUR procedure obtains the highest mean final-window score on all nine maps. The effect of removing a stage is task dependent: *Relative Only* remains competitive on several maps, including `2s3z`, whereas *Absolute Only* performs strongly on `8m` but fails on multiple other scenarios. The checkpoint-level curves in Section I further show that complete BLUR reaches higher observed win rates than both stage variants on `MMM2`, `5m_vs_6m`, and `2c_vs_64zg`. Together, these results support the complete staged training procedure while indicating that the contribution of each stage varies across tasks.

For the reward-model ablations, replacing the recurrent encoder with an MLP consistently reduces the reported mean, although MLP+VaDE remains competitive on several maps. Replacing VaDE with direct regression or KDE produces larger degradation, with BLUR-Direct and BLUR-KDE attaining zero mean performance on several scenarios. These results support the complete combination of recurrent trajectory encoding and VaDE-based latent utility estimation used by BLUR. Small mean differences should be interpreted cautiously in the absence of significance tests.

RQ3. We examine backbone compatibility through BLUR's common learned reward interface. Table 1 shows instantiations with QMIX, MAPPO, DQN, and PPO; backbone-wise curves are provided in Section I. Across the displayed SMAC tasks, MAPPO curves broadly track their handcrafted-reward counterparts, with competitive final-window means in most matched-backbone comparisons.

On `highway-v0`, BLUR reports mean final-window success rates of 0.960 with DQN and 0.940 with PPO. The corresponding handcrafted-reward references report 0.240 and 0.234. On `roundabout-v0`, BLUR reports 0.682 with DQN and 0.882 with PPO, compared with 0.704 and 0.823, respectively. These results show compatibility with the tested backbones in cooperative MARL and two driving tasks, providing initial evidence beyond cooperative MARL rather than broad cross-domain generalization.

RQ4. Figure 6 qualitatively examines utility-related structure through t-SNE Van der Maaten & Hinton (2008) on `8m` at three representative policy states: Early (unconverged Relative Utility Bootstrap with negligible success), Intermediate (Absolute Policy Optimization with non-trivial success), and Converged (late Absolute Policy Optimization with high performance). Their observed test win rates are 0.00, 0.52, and 0.99. Coloring by cluster assignment, episode-level performance, timestep ratio, win/loss outcome, and total policy reward shows organization that becomes more distinguishable in some regions. This qualitative visualization does not establish a unique or causal utility representation, ground-truth temporal credit, or a return decomposition.

Further implementation details, visualizations, and discussion are provided in Sections C, I, and J.

5 Conclusion

We introduced **BLUR**, which reframes dense reward design as learning context-dependent behavior–utility associations from coarse trajectory-level evaluation rather than manually specifying local utility relations. BLUR uses policy-relative supervision to bootstrap learning under failure-dominated experience, then learns a separate baseline-independent reward model whose fixed shaping semantics are used for final policy optimization. The LSTM–VaDE model associates recurring contextual behavioral patterns with trajectory-performance statistics, without assuming a causal or return-preserving utility decomposition.

Experiments on SMAC and HighwayEnv demonstrate compatibility with representative off-policy and on-policy backbones, while training-stage and reward-model ablations support the main components of the framework. Overall, the results suggest that trajectory-performance-supervised utility learning can serve as a practical alternative to handcrafted dense reward specification, while retaining an explicit task-level prior and leaving broader generalization and matched-interaction evaluation for future work.

AI USE STATEMENT

Generative AI tools were used for language polishing and coding assistance. All AI-assisted text and code were reviewed by the authors, who take responsibility for the final paper and implementation.

ETHICS STATEMENT

This work studies reinforcement-learning methodology in simulated environments and does not involve human participants or personally identifiable data.

REPRODUCIBILITY STATEMENT

The main text defines the reward model and three-stage training procedure. The appendix provides the complete algorithm, trajectory-performance construction, optimizer-specific interfaces, experimental configurations, evaluation protocol, and implementation details. The submitted code supplement contains the implementation used for the reported experiments. All reported runs use clean initialization.

REFERENCES

Jose A Arjona-Medina, Michael Gillhofer, Michael Widrich, Thomas Unterthiner, Johannes Brandstetter, and Sepp Hochreiter. Rudder: Return decomposition for delayed rewards. *Advances in Neural Information Processing Systems*, 32, 2019.

Daniel S Bernstein, Robert Givan, Neil Immerman, and Shlomo Zilberstein. The complexity of decentralized control of markov decision processes. *Mathematics of operations research*, 27(4): 819–840, 2002.

Daniel Brown, Wonjoon Goo, Prabhat Nagarajan, and Scott Niekum. Extrapolating beyond suboptimal demonstrations via inverse reinforcement learning from observations. In *International conference on machine learning*, pp. 783–792. PMLR, 2019.

Daniel S Brown, Wonjoon Goo, and Scott Niekum. Better-than-demonstrator imitation learning via automatically-ranked demonstrations. In *Conference on robot learning*, pp. 330–359. PMLR, 2020.

S. Chen, Z. Zhang, Y. Yang, and Y. Du. Stas: Spatial-temporal return decomposition for solving sparse rewards problems in multi-agent reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38(16), pp. 17337–17345, 2024.

Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. *Advances in neural information processing systems*, 30, 2017.

Christoph Dann, Yishay Mansour, and Mehryar Mohri. Reinforcement learning can be more efficient with multiple rewards. In *International Conference on Machine Learning*, pp. 6948–6967. PMLR, 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.

Justin Fu, Katie Luo, and Sergey Levine. Learning robust rewards with adversarial inverse reinforcement learning. *arXiv preprint arXiv:1710.11248*, 2017.

Alex Graves. Long short-term memory. *Supervised sequence labelling with recurrent neural networks*, pp. 37–45, 2012.

Abhishek Gupta, Aldo Pacchiano, Yuexiang Zhai, Sham Kakade, and Sergey Levine. Unpacking reward shaping: Understanding the benefits of reward engineering on sample complexity. *Advances in Neural Information Processing Systems*, 35:15281–15295, 2022.

Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. *Advances in neural information processing systems*, 29, 2016.

Xiao Hu, Jianxiong Li, Xianyuan Zhan, Qing-Shan Jia, and Ya-Qin Zhang. Query-policy misalignment in preference-based reinforcement learning. In *International Conference on Learning Representations*, volume 2024, pp. 45579–45603, 2024.

Borja Ibarz, Jan Leike, Tobias Pohlen, Geoffrey Irving, Shane Legg, and Dario Amodei. Reward learning from human preferences and demonstrations in atari. *Advances in neural information processing systems*, 31, 2018.

Max Jaderberg, Volodymyr Mnih, Wojciech Marian Czarnecki, Tom Schaul, Joel Z Leibo, David Silver, and Koray Kavukcuoglu. Reinforcement learning with unsupervised auxiliary tasks. *arXiv preprint arXiv:1611.05397*, 2016.

Zhuxi Jiang, Yin Zheng, Huachun Tan, Bangsheng Tang, and Hanning Zhou. Variational deep embedding: An unsupervised and generative approach to clustering. *arXiv preprint arXiv:1611.05148*, 2016.

Changyeon Kim, Jongjin Park, Jinwoo Shin, Honglak Lee, Pieter Abbeel, and Kimin Lee. Preference transformer: Modeling human preferences using transformers for rl. *arXiv preprint arXiv:2303.00957*, 2023.

Kimin Lee, Laura Smith, and Pieter Abbeel. Pebble: Feedback-efficient interactive reinforcement learning via relabeling experience and unsupervised pre-training. *arXiv preprint arXiv:2106.05091*, 2021.

Edouard Leurent et al. An environment for autonomous driving decision-making, 2018.

H. Lin, H. Wu, J. Zhang, Y. Sun, J. Ye, and Y. Yu. Episodic return decomposition by difference of implicitly assigned sub-trajectory reward. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38(12), pp. 13808–13816, 2024.

Runze Liu, Fengshuo Bai, Yali Du, and Yaodong Yang. Meta-reward-net: Implicitly differentiable reward learning for preference-based reinforcement learning. *Advances in Neural Information Processing Systems*, 35:22270–22284, 2022.

Runze Liu, Yali Du, Fengshuo Bai, Jiafei Lyu, and Xiu Li. Pearl: Zero-shot cross-task preference alignment and robust reward learning for robotic manipulation. *arXiv preprint arXiv:2306.03615*, 2023.

Haozhe Ma, Zhengding Luo, Thanh Vinh Vo, Kuankuan Sima, and Tze-Yun Leong. Highly efficient self-adaptive reward shaping for reinforcement learning. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=QOfWubPhdS.

Kota Minoshima and Sachiyo Arai. Autonomous reward shaping via self-generated trajectories for sparse-reward reinforcement learning. *IEEE Access*, 2026.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

Andrew Y Ng, Daishi Harada, and Stuart Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *Icml*, volume 99, pp. 278–287. Citeseer, 1999.

Andrew Y Ng, Stuart Russell, et al. Algorithms for inverse reinforcement learning. In *Icml*, volume 1, pp. 2, 2000.

Junhyuk Oh, Yijie Guo, Satinder Singh, and Honglak Lee. Self-imitation learning. In *International conference on machine learning*, pp. 3878–3887. PMLR, 2018.

Frans A Oliehoek, Christopher Amato, et al. *A concise introduction to decentralized POMDPs*, volume 1. Springer, 2016.

Jongjin Park, Younggyo Seo, Jinwoo Shin, Honglak Lee, Pieter Abbeel, and Kimin Lee. Surf: Semi-supervised reward learning with data augmentation for feedback-efficient preference-based reinforcement learning. *arXiv preprint arXiv:2203.10050*, 2022.

Vihang P Patil, Markus Hofmarcher, Marius-Constantin Dinu, Matthias Dorfer, Patrick M Blies, Johannes Brandstetter, Jose A Arjona-Medina, and Sepp Hochreiter. Align-rudder: Learning from few demonstrations by reward redistribution. *arXiv preprint arXiv:2009.14108*, 2020.

Tabish Rashid, Mikayel Samvelyan, Christian Schroeder De Witt, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. Qmix: Monotonic value function factorisation for deep multi-agent reinforcement learning. In *Proceedings of the 35th International Conference on Machine Learning*, pp. 4295–4304, 2018.

Z. Ren, R. Guo, Y. Zhou, and J. Peng. Learning long-term reward redistribution via randomized return decomposition. In *International Conference on Learning Representations*, 2022.

Mikayel Samvelyan, Tabish Rashid, Christian Schroeder De Witt, Gregory Farquhar, Nantas Nardelli, Tim GJ Rudner, Chia-Man Hung, Philip HS Torr, Jakob Foerster, and Shimon Whiteson. The starcraft multi-agent challenge. *arXiv preprint arXiv:1902.04043*, 2019.

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

Jonathan Sorg, Richard L Lewis, and Satinder Singh. Reward design via online gradient ascent. *Advances in Neural Information Processing Systems*, 23, 2010.

Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008.

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

Brian D Ziebart, Andrew L Maas, J Andrew Bagnell, Anind K Dey, et al. Maximum entropy inverse reinforcement learning. In *Aaai*, volume 8, pp. 1433–1438. Chicago, IL, USA, 2008.

A Appendix

B Preliminary

B.1 Variational Deep Embedding

Variational Deep Embedding (VaDE) combines a variational autoencoder with a Gaussian-mixture prior for probabilistic clustering. In BLUR, VaDE receives an LSTM hidden representation $\mathbf{h}$ and encodes it into a latent variable $\mathbf{z}$ through

$$q_\phi\left(\mathbf{z} \mid \mathbf{h}\right) = \mathcal{N}\left(\boldsymbol{\mu}_\phi(\mathbf{h}), \operatorname{diag}\left(\boldsymbol{\sigma}^2_\phi(\mathbf{h})\right)\right). \tag{14}$$

The decoder reconstructs $\mathbf{h}$ from $\mathbf{z}$, while the Gaussian-mixture prior provides posterior cluster responsibilities. Let π_k, $\boldsymbol{\mu}_k$, and $\boldsymbol{\Sigma}_k$ denote the mixture weight, mean, and covariance of component k. The cluster assignment used by BLUR is

$$p(c = k \mid \mathbf{z}) = \frac{\pi_k \mathcal{N}\left(\mathbf{z}; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\right)}{\sum_{j=1}^{K} \pi_j \mathcal{N}\left(\mathbf{z}; \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j\right)}. \tag{15}$$

BLUR uses an adapted VaDE-based reconstruction and mixture-regularization objective

$$\mathcal{L}_{\mathrm{VaDE}} = \lambda_{\mathrm{rec}} \mathcal{L}_{\mathrm{rec}} + \lambda_{\mathrm{KL}} \mathcal{L}_{\mathrm{KL}}, \tag{16}$$

where $\mathcal{L}_{\mathrm{rec}}$ reconstructs the input representation and $\mathcal{L}_{\mathrm{KL}}$ denotes responsibility-weighted regularization toward the Gaussian-mixture components. This objective retains the VaDE mixture latent structure but is adapted to BLUR's sequential reward-learning interface rather than presented as the exact canonical VaDE ELBO. BLUR-specific optimization and utility estimation are detailed in Sections C.3 and C.4.

B.2 QMIX

QMIX is a value-decomposition method for cooperative multi-agent reinforcement learning under centralized training and decentralized execution. Each agent learns an individual action-value function $Q_i(o_t^i, a_t^i)$, while a centralized mixing network combines the individual values into a joint action-value function:

$$Q_{\text{tot}} = f_{\text{mix}}\left(Q_1, \ldots, Q_N; s_t\right). \tag{17}$$

The mixing network satisfies the monotonicity constraint

$$\frac{\partial Q_{\text{tot}}}{\partial Q_i} \geq 0, \qquad i = 1, \ldots, N, \tag{18}$$

which makes decentralized greedy action selection consistent with the joint greedy action. QMIX is trained off-policy from replayed trajectories using temporal-difference learning. In BLUR, QMIX serves only as the policy optimizer: BLUR supplies the stage-specific learned reward to the temporal-difference target without altering the value-decomposition architecture.

For a replay batch $\mathcal{B}$, the QMIX temporal-difference objective is

$$\mathcal{L}_{\text{QMIX}} = \mathbb{E}_{\mathcal{B}}\left[\left(y_t - Q_{\text{tot}}(\boldsymbol{\eta}_t, \mathbf{a}_t)\right)^2\right], \tag{19}$$

where

$$y_t = r_t^{\text{stage}} + \gamma(1 - d_t) \max_{\mathbf{a}'} Q_{\text{tot}}^{-}\left(\boldsymbol{\eta}_{t+1}, \mathbf{a}'\right). \tag{20}$$

Here, $\boldsymbol{\eta}_t$ denotes the collection of agent action–observation histories up to time step t, while $d_t \in \{0, 1\}$ is the terminal indicator. The quantity r_t^{stage} denotes the BLUR reward used in the current policy-optimization stage, namely r_t^{boot} during Relative Utility Bootstrap and r_t^{final} during Absolute Policy Optimization.

C Additional Reward-Model and Training Details

This section separates the shared BLUR estimator from the domain-specific training procedures used in the final implementations. BLUR does not consume the handcrafted dense environment reward. Trajectory performance supervises the reward model, while the policy reward uses learned shaping and the sparse terminal task outcome according to the interfaces below.

C.1 Reward-Model Architecture

The figure below illustrates the architecture used by the Relative Utility Model and the Absolute Reward Model. The raster schematic labels its contextual input as O_t; this corresponds to the generic input $\mathbf{x}_t$ in the formulation. The labels i and j denote variable cluster indices.

C.2 Sequence Pooling and Variable-Length Trajectories

The auxiliary performance-prediction head uses masked mean pooling over valid LSTM hidden states:

$$\text{Pool}(\mathbf{h}_{1:T_{\max}}) = \frac{\sum_{t=1}^{T_{\max}} m_t \mathbf{h}_t}{\sum_{t=1}^{T_{\max}} m_t}, \tag{21}$$

where $m_t \in \{0, 1\}$ distinguishes valid timesteps from padding. The SMAC implementation supplies sequence lengths to this operation; the HighwayEnv implementation additionally packs variable-length sequences before the LSTM.

C.3 Sequential Encoder and VaDE Optimization

Both implementations first optimize the temporal encoder and performance head under trajectory-level mean-squared error. Hidden representations are then detached before VaDE fitting, so VaDE gradients do not propagate into the LSTM. For hidden representation $\mathbf{h}$, VaDE forms $q_\phi(\mathbf{z} \mid \mathbf{h})$

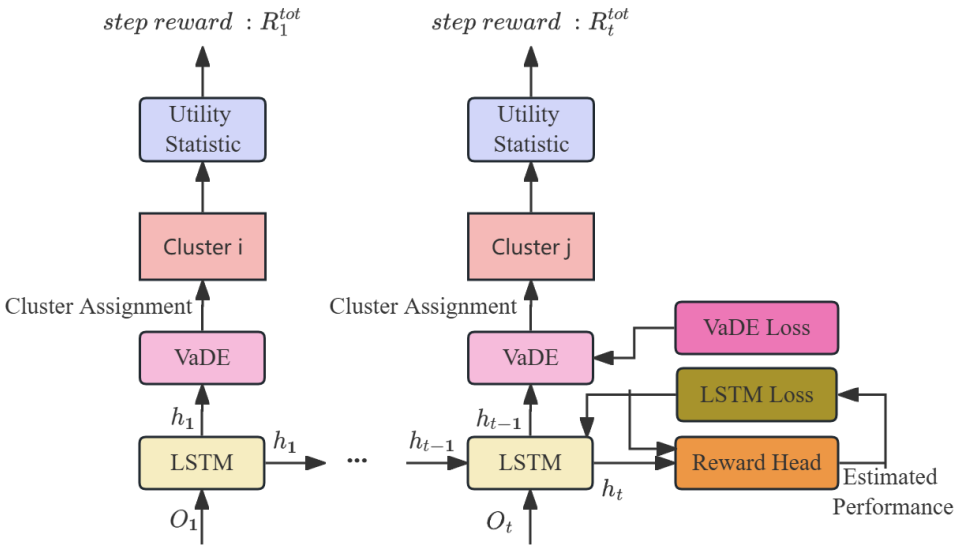

Figure 2: Reward-model architecture. The schematic input O_t corresponds to $\mathbf{x}_t$ in the formulation, and i, j are latent-cluster indices.

with learned mean and log variance, decodes $\mathbf{z}$ back to the hidden space, and regularizes it toward a learned Gaussian-mixture prior. The objective has the form

$$\mathcal{L}_{\text{VaDE}} = \lambda_{\text{rec}}\mathcal{L}_{\text{rec}} + \lambda_{\text{KL}}\mathcal{L}_{\text{KL}}. \tag{22}$$

The SMAC code uses responsibility-weighted component regularization with the responsibilities detached inside the KL term. The HighwayEnv code includes both the responsibility-weighted Gaussian-component KL and the categorical-mixture term. During utility estimation and reward inference, both implementations use deterministic responsibilities evaluated at the posterior mean rather than a sampled latent point.

C.4 Timestep-Level Utility Aggregation and Scaling

Given M trajectories with valid lengths T_m, the raw utility statistic of cluster k is

$$\widetilde{U}_k = \frac{\sum_{m=1}^{M}\sum_{t=1}^{T_m}\gamma_{m,t,k}P(\tau_m)}{\sum_{m=1}^{M}\sum_{t=1}^{T_m}\gamma_{m,t,k} + \varepsilon}. \tag{23}$$

Each valid timestep contributes one posterior-weighted sample; no additional trajectory-length reweighting is applied. The raw statistics are standardized across clusters and clipped to $[-1, 1]$ before reward inference. Thus, $\widetilde{U}_k$ is an association statistic between latent pattern k and trajectory performance, not a causal marginal contribution or a return-preserving decomposition.

D Relative Utility Bootstrap Details

D.1 Relative Labels

For trajectory τ_n, both reported implementations use the unnormalized label

$$P^{\text{rel}}(\tau_n) = P^{\text{abs}}(\tau_n) - b_n, \tag{24}$$

where b_n is the mean of a recent-performance window before inserting the current trajectory. The reported window contains 100 trajectories. The first SMAC trajectory uses baseline zero; HighwayEnv collects an initial set before the first policy update. Both reported implementations use the unnormalized label above.

D.2 SMAC Relative Stage

SMAC does not perform an initial reward-model fit before policy learning. It starts with the raw-relative fallback

$$r_t^{\text{raw}} = \text{clip}\left(\frac{P^{\text{rel}}(\tau)}{T_\tau}, -5, 5\right) + \mathbb{I}[t = T_\tau]\mathbb{I}_{\text{win}}(\tau) + 2\,\mathbb{I}[t = T_\tau]\mathbb{I}[P^{\text{abs}}(\tau) > P_{\text{best}} + 1]. \tag{25}$$

The final term is the configured breakthrough bonus. With probability 0.30, the implementation instead collects a random-policy exploration trajectory for the reward-model dataset without using it for a relative-policy update.

Relative-model fitting first becomes eligible after 500 bootstrap policy updates and is reconsidered every 500 updates using at most the most recent 3000 relative-bootstrap trajectories. Fitting is skipped until the standard deviation of the stored relative labels is at least 0.1. A fitted relative model becomes usable when its reward standard deviation is at least 0.03 and its normalized cluster entropy is at least 0.2. Before first acceptance, the code permits up to three consecutive fit attempts at an eligible update; later updates use one attempt.

After the model is usable, the reported SMAC bootstrap reward is

$$r_t^{\text{boot}} = 0.5 r_t^{\text{raw}} + 0.5 r_t^{\text{rel}}, \tag{26}$$

where r_t^{rel} is the posterior-weighted Relative Utility Model output. Consequently, the raw terminal win and breakthrough components remain present through their 0.5 mixture weight. The relative stage is evaluated every 500 policy updates over 16 episodes and ends when evaluation win rate is at least 0.15 or when $50{,}000$ bootstrap policy updates are reached.

D.3 HighwayEnv Relative Stage

HighwayEnv collects 32 trajectories without policy updates, labels each using the preceding baseline queue, and fits the initial Relative Utility Model before the first relative-policy update. There is no raw-relative fallback. The model is updated after every collected training episode using a sample of at most 64 stored trajectories. The policy reward is

$$r_t^{\text{boot}} = 0.5 r_t^{\text{rel}} + 10\,\mathbb{I}[t = T_\tau]\mathbb{I}_{\text{succ}}(\tau). \tag{27}$$

Evaluation uses 10 episodes every 1000 policy updates. The stage ends when evaluation success rate is at least 0.6 or when $50{,}000$ relative-policy updates are reached.

D.4 Optimizer-Specific Relative Policy Updates

Off-policy optimizers. SMAC/QMIX stores complete trajectories and recomputes the active raw, learned, or mixed relative reward when a trajectory is sampled. HighwayEnv/DQN stores transitions whose episode pointer allows the current reward model to recompute the relevant timestep reward. Thus, policy replay does not store an immutable copy of a learned reward.

On-policy optimizers. MAPPO and PPO update only from fresh rollouts. In SMAC, a pending MAPPO rollout is cleared when the reward source changes. Historical success-replay trajectories are not inserted into either PPO surrogate objective.

D.5 Adaptation to MAPPO

The SMAC MAPPO implementation accumulates eight fresh episodes per policy update. All episodes in one rollout batch use the same active reward source. The final-stage actor and critic are independently initialized rather than carried over from Phase I. Success replay remains off-policy auxiliary data and is not used for MAPPO gradient updates.

E Absolute Utility Learning Details

E.1 Trajectory-Level Performance Construction

BLUR constructs $P^{\mathrm{abs}}(\tau)$ from complete-trajectory task statistics. These statistics encode task knowledge but do not assign rewards to individual transitions.

E.2 Absolute Performance in SMAC

The reported SMAC computation is

$$\begin{aligned} P^{\mathrm{abs}}(\tau) = 100\mathbb{I}_{\mathrm{win}}(\tau) \\ + 10\left[\mathrm{clip}_{[0,1]}\left(\frac{N_{\mathrm{kill}}}{N_{\mathrm{enemy}}}\right) + 0.5\,\mathrm{clip}_{[0,1]}\left(\frac{D_{\mathrm{dealt}}}{H_{\mathrm{enemy}}}\right)\right] \\ - 5\left[\mathrm{clip}_{[0,1]}\left(\frac{N_{\mathrm{death}}}{N_{\mathrm{ally}}}\right) + 0.5\,\mathrm{clip}_{[0,1]}\left(\frac{D_{\mathrm{taken}}}{H_{\mathrm{ally}}}\right)\right]. \end{aligned} \tag{28}$$

Counts and cumulative damage are read from episode information. Denominators use configured or environment-reported totals and are clamped away from zero; if damage fields are unavailable, the damage terms remain zero. This score is used only as reward-model supervision.

E.3 Absolute Performance in HighwayEnv

For both evaluated HighwayEnv tasks, the executed computation is

$$\begin{aligned} P^{\mathrm{abs}}(\tau) = 100\mathbb{I}_{\mathrm{succ}}(\tau) - 100\mathbb{I}_{\mathrm{crash}}(\tau) + 10\bar{v}_{\mathrm{norm}} + 10q_{\mathrm{progress}} \\ - 5q_{\mathrm{offroad}} - q_{\mathrm{lanechange}}, \end{aligned} \tag{29}$$

where $\bar{v}_{\mathrm{norm}} = \mathrm{clip}(\bar{v}/30, 0, 1)$, $q_{\mathrm{progress}} = \mathrm{clip}(T/200, 0, 1)$, $q_{\mathrm{offroad}} = \mathrm{clip}(N_{\mathrm{offroad}}/T, 0, 1)$, and $q_{\mathrm{lanechange}} = \mathrm{clip}(N_{\mathrm{lanechange}}/\max(1, T/10), 0, 1)$. Success is read from the native terminal environment information rather than inferred from these statistics.

E.4 SMAC Absolute-Model Fitting and Readiness

The SMAC Absolute Reward Model is initialized independently of the relative model. Each fitting call draws at most 512 trajectories from the trajectory buffer with target proportions 20% successful, 20% top-performance failures, 20% diversity-ranked failures, 15% random-exploration data, 15% older relative-bootstrap data, and 10% recent absolute-policy data. Overlapping trajectories are deduplicated and missing category slots are backfilled uniformly from the remaining buffer. This sampler is distinct from the policy Success Replay Buffer.

The fitted model is accepted only if all of the following hold:

$$\text{sign match} \geq 0.55, \qquad 10^{-3} \leq \mathrm{Std}(R_\tau) \leq 10, \qquad H^{\mathrm{norm}}_{\mathrm{cluster}} \geq 0.20. \tag{30}$$

Here, R_τ is the sum of inferred timestep rewards used by the readiness check, including the configured terminal win bonus. If the gate fails, one random-policy exploration trajectory is added and the model is fitted again.

After acceptance, the current policy performs a Success Replay warm start. The buffer capacity is 5000 episodes; collection targets up to 3000 winning trajectories with at most $10{,}000$ attempts. The buffer also stores near-success episodes, defined in code as non-winning trajectories with $P^{\mathrm{abs}}(\tau) > 0$. These do not count toward the winning-trajectory target. During sampling, the requested composition is 80% wins and 20% near successes, with fallback to the available pool.

E.5 HighwayEnv Absolute-Model Fitting

HighwayEnv initializes a fresh Absolute Reward Model and freezes the relative policy as its collector. Each fit uses at most 64 trajectories. When the success buffer is nonempty, half of the requested slots are sampled with replacement from successful trajectories and the other half from the general trajectory buffer; otherwise all samples come from the general buffer. The reported transition condition requires at least 32 stored trajectories and one absolute-model fit round. The accepted absolute model is then frozen.

F Absolute Policy Optimization Details

F.1 SMAC Policy and Reward Updates

SMAC independently initializes the final QMIX or MAPPO policy before absolute policy optimization. The Absolute Reward Model remains frozen throughout this stage. The absolute reward is the posterior-weighted cluster utility plus a terminal win bonus of 10. QMIX may select a Success Replay batch with probability

$$p_{\mathrm{succ}}(u) = \max\left(0.1, 0.5\exp(-u/50{,}000)\right), \tag{31}$$

where u is the number of absolute-stage policy updates. A selected Success Replay batch is drawn as an episode batch rather than mixed into every ordinary minibatch. MAPPO always uses fresh on-policy rollouts.

At an absolute-stage validation checkpoint with win rate greater than 0.1, the implementation attempts to refill Success Replay with up to 500 additional wins, using at most 3000 collection episodes.

F.2 HighwayEnv Policy and Reward Updates

HighwayEnv freezes the Absolute Reward Model, initializes a fresh target DQN/PPO policy, and gives DQN a fresh final replay buffer. Its final reward is

$$r_t^{\mathrm{final}} = 0.5 r_t^{\mathrm{abs}} + 10\,\mathbb{I}[t = T_\tau]\mathbb{I}_{\mathrm{succ}}(\tau). \tag{32}$$

For each DQN gradient update, a Success Replay transition batch is selected with probability 0.5 when successful data are available; otherwise the batch comes from the final replay buffer. Highway Success Replay contains only terminal-success trajectories and has capacity 5000. PPO uses only the fresh current-policy episode and never reuses historical success trajectories.

F.3 Budget, Stopping, and Evaluation Accounting

SMAC uses implementation-stage caps of $50{,}000$ updates for Relative Utility Bootstrap and $500{,}000$ updates for Absolute Policy Optimization. SMAC evaluates every 500 policy updates using 16 episodes. Reported learning curves and final-window summaries are truncated to the task–backbone–comparison horizon $H_{e,b,c}$ jointly supported by the corresponding methods and seeds.

For HighwayEnv, the relative stage is capped at $50{,}000$ policy updates. Reported DQN and PPO training extents are determined separately from their valid run support through $H_{e,b,c}$ rather than a universal total-update cap. HighwayEnv evaluates every 1000 policy updates using 10 episodes. DQN counts one gradient update per collected timestep and PPO counts one update per fresh episode rollout. In both domains, internal stage controls do not define the cross-method reporting horizon. Absolute-model fitting and auxiliary collection consume environment interactions but not policy updates, so common global-update support does not match total environment interactions or wall-clock computation.

G Algorithm

Algorithm 1: Executed SMAC/QMIX BLUR Procedure

Require: Environment $\mathcal{E}$; trajectory performance P^{abs}; bootstrap cap $50{,}000$; absolute-policy cap $500{,}000$; evaluation interval 500 updates; 16 evaluation episodes

Ensure: Trained target policy π_{abs} and frozen absolute reward model $\mathcal{R}_{\theta_{\mathrm{abs}}}$

1: Initialize relative policy π_{rel}, temporary relative model $\mathcal{R}_{\theta_{\mathrm{rel}}}$, general replay $\mathcal{D}_{\mathrm{replay}}$, trajectory buffer $\mathcal{D}_{\mathrm{traj}}$, Success Replay $\mathcal{D}_{\mathrm{succ}}$, and a length-100 performance queue
Stage I: Relative Utility Bootstrap
2: **while** evaluation win rate < 0.15 and bootstrap updates $< 50{,}000$ **do**

With probability 0.30, collect a random-policy exploration trajectory for $\mathcal{D}_{\mathrm{traj}}$ only
Otherwise collect $\tau \sim \pi_{\mathrm{rel}}$, compute $P^{\mathrm{rel}}(\tau) = P^{\mathrm{abs}}(\tau) - \mathrm{Mean}(\mathcal{Q})$, and store τ in both general buffers
Compute the raw fallback r^{raw} by distributing $P^{\mathrm{rel}}(\tau)$ across valid timesteps, clipping to $[-5, 5]$, and adding the configured terminal win and breakthrough terms
if $\mathcal{R}_{\theta_{\mathrm{rel}}}$ has passed its quality gate **then**
    $r^{\mathrm{boot}} \leftarrow 0.5r^{\mathrm{raw}} + 0.5\mathcal{R}_{\theta_{\mathrm{rel}}}(\tau)$
else
    $r^{\mathrm{boot}} \leftarrow r^{\mathrm{raw}}$
end if
Sample/fill a QMIX episode batch, recompute its active relative rewards, and update π_{rel}
Append $P^{\mathrm{abs}}(\tau)$ to $\mathcal{Q}$
if bootstrap updates are at least 500 and divisible by 500 **then**
    Fit the relative encoder and VaDE on at most the latest 3000 relative trajectories when label standard deviation is at least 0.1
    Mark the model usable only if reward standard deviation is at least 0.03 and normalized cluster entropy is at least 0.2
end if
Every 500 policy updates, evaluate over 16 episodes
end while
Freeze π_{rel} as the Stage-II collector and stop using $\mathcal{R}_{\theta_{\mathrm{rel}}}$
Stage II: Absolute Utility Learning
Initialize $\mathcal{R}_{\theta_{\mathrm{abs}}}$ from scratch
repeat
    Draw at most 512 trajectories from $\mathcal{D}_{\mathrm{traj}}$ with target ratios $20/20/20/15/15/10$ for success, top failure, diverse failure, exploration, bootstrap, and recent absolute-policy data
    Train the absolute encoder, detach hidden representations, fit VaDE, and estimate absolute cluster utilities
    Check sign match ≥ 0.55, episode-return standard deviation in $[10^{-3}, 10]$, and normalized cluster entropy ≥ 0.20
    if the gate fails **then**
        Collect one random-policy exploration trajectory and add it to $\mathcal{D}_{\mathrm{traj}}$
    end if
until the absolute reward model passes the gate
Freeze $\mathcal{R}_{\theta_{\mathrm{abs}}}$ for final policy optimization
Using π_{rel}, collect up to 3000 winning trajectories for $\mathcal{D}_{\mathrm{succ}}$, stopping after $10{,}000$ attempts, and retain non-winning trajectories with $P^{\mathrm{abs}}(\tau) > 0$ as near successes
Stage III: Absolute Policy Optimization
Independently initialize the final target policy π_{abs}
while absolute-policy updates $< 500{,}000$ **do**
    Let $p_{\mathrm{succ}}(u) = \max(0.1, 0.5e^{-u/50{,}000})$
    if a Bernoulli draw with probability $p_{\mathrm{succ}}(u)$ succeeds **then**
        Draw the QMIX episode batch from $\mathcal{D}_{\mathrm{succ}}$
    else
        Collect a trajectory from π_{abs}, add it to the replay and trajectory buffers, and form a batch from general replay
    end if
    Recompute r_t^{abs} and add terminal win bonus 10
    Update π_{abs} with the resulting reward batch
    At validation win rate > 0.1, attempt a Success Replay refill of up to 500 wins with at most 3000 collection episodes
end while
return $\pi_{\mathrm{abs}}, \mathcal{R}_{\theta_{\mathrm{abs}}}$

For SMAC/MAPPO, the final actor and critic are also independently initialized, policy batches are fresh eight-episode rollouts, and historical Success Replay is not used in the surrogate objective. HighwayEnv follows the same semantic stages but uses the domain-specific initialization, stopping, sampling, replay, and budget-accounting rules in Section C. Reported curves and final-window results use common global-update support within each matched task–backbone comparison; the stage caps above are implementation limits.

H Detailed Experimental Setup

This appendix provides additional details on the environments, baselines, evaluation metrics, implementation, and statistical reporting used in our experiments.

H.1 Environments

SMAC. Our primary experiments are conducted on the StarCraft Multi-Agent Challenge (SMAC). We consider scenarios spanning different levels of difficulty, team sizes, unit compositions, and degrees of coordination. The evaluated maps include `3m`, `8m`, `2s3z`, `2s_vs_1sc`, `1c3s5z`, `5m_vs_6m`, `2c_vs_64zg`, `corridor`, and `MMM2`.

Each agent receives its local observation and executes actions under decentralized execution. The reward model is used only during centralized training. Its contextual input $\mathbf{x}_t$ is instantiated as the global state when centralized state information is available and as the environment observation otherwise. Agent policies continue to select actions from their local observations during decentralized execution.

The trajectory-level performance signal follows the executed combat-statistic formula in Section E.2. The reported configuration uses a win coefficient of 100, kill/death coefficients of $10/5$, and damage multipliers of $0.5/0.5$.

HighwayEnv. We evaluate single-agent applicability on `highway-v0` and `roundabout-v0`. BLUR is instantiated with DQN and PPO while retaining the corresponding value-based and actor–critic policy architectures. For both tasks, evaluation success is read directly from the success information returned by the environment. We do not introduce an additional manually designed success classifier or post-processing rule. The reported test success rate is the fraction of evaluation episodes labeled successful by the environment. The reward-model supervision $P^{\mathrm{abs}}(\tau)$ combines terminal success, crash, normalized average speed, episode progress, off-road ratio, and normalized lane-change frequency as specified in Section E.3.

H.2 Compared Methods

Handcrafted Reward. The policy is trained with the original manually designed dense step-wise environment reward. This baseline serves as a strong performance reference: BLUR aims to approach or surpass its policy-learning performance without requiring the corresponding manually engineered dense transition-level reward specification.

SASR. We independently reimplement SASR following the published formulation. The SMAC experiments adapt it to the QMIX interface, whereas the HighwayEnv results reported in Table 1 use PPO. We retain its success/failure-conditioned reward-learning mechanism. Successful and failed episodes are identified using the environment-provided terminal outcome, and their state samples are stored in separate buffers. The learned SASR reward is recomputed during QMIX replay updates according to the implementation provided in the supplementary code.

The original SASR formulation adds its learned shaping reward to the environment reward. However, the native SMAC reward contains dense handcrafted step-wise information, whereas BLUR is evaluated without that dense component. We use

$$r_t^{\text{SASR-fair}} = r_t^{\text{term}} + \lambda_{\text{SASR}} \widehat{r}_t^{\text{SASR}}, \tag{33}$$

where r_t^{term} is the environment-provided terminal win/loss reward and is zero at nonterminal steps. The reported SASR configuration uses dense environment weight 0, terminal win/loss weight 1, and $\lambda_{\text{SASR}} = 0.6$. The sparse component therefore preserves its native scale while SASR supplies the learned shaping term. Under our evaluation protocol, SASR attains zero test win rate on several evaluated scenarios; these zero-valued results are retained in the reported comparisons.

BLUR. The complete method uses relative utility bootstrapping, absolute utility learning, and final absolute policy optimization. Its reported reward model uses an LSTM trajectory encoder followed by VaDE-based latent utility modeling.

Stage ablations. We evaluate:

- **Relative Only**, which retains relative utility learning without transitioning to a newly learned absolute reward semantics;
- **Absolute Only**, which removes relative bootstrapping and directly learns the absolute reward model from the initially available trajectory distribution.

Reward-model ablations. We compare the complete **BLUR** reward model, which combines an LSTM trajectory encoder with VaDE-based latent utility estimation, against:

- **MLP+VaDE**, which replaces the recurrent temporal encoder with an MLP while retaining VaDE-based latent utility estimation;
- **BLUR-Direct**, which retains the LSTM encoder but replaces VaDE-based latent utility estimation with direct utility regression;
- **BLUR-KDE**, which retains the LSTM encoder but replaces VaDE with Gaussian-kernel KDE utility estimation.

These variants keep the staged BLUR training procedure unchanged unless the stage itself is the object of ablation.

H.3 Policy Backbones

The SMAC experiments use QMIX and MAPPO as representative off-policy and on-policy backbones. The HighwayEnv experiments use DQN and PPO as the corresponding single-agent backbones.

Within each comparison, BLUR and the handcrafted-reward baseline use the same policy architecture and optimizer family. Reward source, stage-specific data management, and method-specific reward-model components differ. Off-policy trajectory reuse and reward recomputation are described in Section D.4. For MAPPO, policy updates use fresh on-policy rollouts under the reward model fixed for the current update phase; see Section D.5.

H.4 Evaluation Metrics and Statistical Aggregation

SMAC metrics. The primary metric is test win rate, computed over evaluation episodes without exploration. We additionally record average trajectory performance, enemy kills, allied deaths, and environment return for diagnostic analyses.

For task or environment e, policy backbone b, comparison set c, reward method $m \in \mathcal{M}_{b,c}$, and seed s, let $U_{\max}(e, b, c, m, s)$ be the largest global policy-update count with valid evaluation data. The comparison-specific reporting horizon is

$$H_{e,b,c} = \min_{m \in \mathcal{M}_{b,c}, s} U_{\max}(e, b, c, m, s), \tag{34}$$

the largest endpoint jointly supported by all methods and seeds within the same task, backbone, and comparison set. The index c is necessary because the main comparison, training-stage ablation, and reward-model ablation can contain different method sets even when e and b are unchanged.

For each (e, b, c), the common reporting grid covers only the jointly supported range and satisfies

$$u_{e,b,c,J_{e,b,c}} = H_{e,b,c}, \qquad u_{e,b,c,j+1} - u_{e,b,c,j} = 500 \tag{35}$$

Evaluation values are linearly interpolated between adjacent logged evaluation checkpoints onto this 500-policy-update grid. Let $w_{e,b,c,m,s,j} \in [0, 1]$ denote the resulting test win rate or success rate at grid point $u_{e,b,c,j}$. The seed-level final-window score is

$$F_{e,b,c,m,s} = \frac{1}{L} \sum_{j=J_{e,b,c}-L+1}^{J_{e,b,c}} w_{e,b,c,m,s,j}, \qquad L = 10, \tag{36}$$

so the window contains the final ten aligned grid points rather than the last ten records from a method's raw evaluation schedule. With 500-update spacing, this window summarizes the final ten

points of the common reporting grid. These grid-point values are averaged within each seed before cross-seed aggregation. The cross-seed mean is

$$\overline{F}_{e,b,c,m} = \frac{1}{S}\sum_{s=1}^{S} F_{e,b,c,m,s}, \qquad S = 5, \tag{37}$$

and the reported sample standard deviation is

$$\sigma_{e,b,c,m} = \sqrt{\frac{1}{S-1}\sum_{s=1}^{S}\left(F_{e,b,c,m,s} - \overline{F}_{e,b,c,m}\right)^2}. \tag{38}$$

Each main-table entry reports $\overline{F}_{e,b,c,m} \pm \sigma_{e,b,c,m}$. The standard deviation therefore reflects variation across the five seed-level final-window averages, not temporal variation among the ten grid points within an individual run. Final-window aggregation avoids selecting a single best checkpoint and reduces sensitivity to isolated evaluation spikes; the ten interpolated grid values are not treated as independent observations. The aligned grid contains at least $L = 10$ points for every reported comparison; "–" denotes an unavailable method–task combination. Table 1 uses no boldface performance values because its rows contain multiple backbones. Table 2 uses boldface for the highest reported mean within each QMIX-only row; this does not indicate statistical significance.

For the main comparison, SMAC with QMIX contains Handcrafted, SASR, and BLUR; SMAC with MAPPO and HighwayEnv with DQN contain Handcrafted and BLUR; and HighwayEnv with PPO contains Handcrafted, SASR, and BLUR. The QMIX training-stage comparison contains BLUR, Relative Only, and Absolute Only, whereas the QMIX reward-model comparison contains BLUR, MLP+VaDE, BLUR-Direct, and BLUR-KDE. Each comparison set is aligned separately.

HighwayEnv metrics. The primary reported HighwayEnv metric is test success rate.

H.5 Comparison-Specific Global Reporting Horizons

Table 3 lists selected $H_{e,b,c}$ values recoverable from the displayed learning-curve comparisons; it is not intended to enumerate every comparison in Tables 1 and 2. Each listed value is jointly supported by the methods and seeds within its task, backbone, and comparison set. These horizons are distinct from internal optimizer or stage caps and do not imply matched total environment interaction.

Table 3: Selected comparison-specific common-support horizons $H_{e,b,c}$ recoverable from the displayed learning curves.

Task	Backbone	$H_{e,b,c}$	Comparison set
1c3s5z	QMIX	50,000	RQ1 main
2c_vs_64zg	QMIX	98,000	RQ1 main
2s3z	QMIX	47,000	RQ1 main
2s_vs_1sc	QMIX	43,000	RQ1 main
3m	QMIX	34,000	RQ1 main
5m_vs_6m	QMIX	281,000	RQ1 main
8m	QMIX	50,000	RQ1 main
corridor	QMIX	39,000	RQ1 main
MMM2	QMIX	327,500	RQ2 training-stage
2c_vs_64zg	MAPPO	100,000	RQ3 backbone
3m	MAPPO	85,000	RQ3 backbone
8m	MAPPO	51,000	RQ3 backbone
MMM2	MAPPO	110,500	RQ3 backbone
highway-v0	DQN	33,000	RQ3 backbone
roundabout-v0	DQN	500,000	RQ3 backbone

H.6 Training and Evaluation Protocol

All reported runs use clean initialization.

SMAC. BLUR is evaluated every 500 policy updates using 16 episodes without exploratory action noise. Relative Utility Bootstrap stops at evaluation win rate at least 0.15 or 50,000 bootstrap

updates. Absolute Policy Optimization has an implementation cap of $500{,}000$ updates. The final QMIX or MAPPO policy is independently initialized, and the accepted Absolute Reward Model remains frozen throughout its training. Absolute reward fitting, readiness collection, Success Replay warm start, and refill do not increment a policy-update counter. Reported curves and final-window summaries use the comparison-specific horizon $H_{e,b,c}$ jointly supported by the corresponding methods and seeds.

HighwayEnv. Evaluation occurs every 1000 policy updates using 10 episodes. The relative stage is capped at $50{,}000$ updates or success rate at least 0.6; the reported final-training extent is determined from valid run support through $H_{e,b,c}$ rather than a universal total-update cap. Absolute-model fitting performs no policy updates. The accepted model is frozen, and a fresh target DQN/PPO policy is created; DQN also receives a fresh final replay buffer. With `updates_per_episode`=0, DQN performs one gradient update per collected timestep, while PPO records one policy update per fresh episode rollout.

SASR. The independent SASR implementation evaluates every 1000 training episodes using 16 evaluation episodes. Its reported limits are $20{,}000$ training episodes and $2{,}000{,}000$ environment steps. This raw evaluation schedule is distinct from the comparison procedure: SASR records are linearly interpolated onto the common grid only for their corresponding backbone comparison—QMIX in SMAC and PPO in HighwayEnv—and evaluated using the final $L = 10$ aligned grid points within $H_{e,b,c}$. SASR therefore does not restrict MAPPO or DQN support.

These protocols do not match total environment interactions or wall-clock computation. They also use different meanings of one policy update across QMIX, MAPPO, DQN, and PPO. Detailed stage accounting is provided in Section F.3.

H.7 Reward-Model Configuration

Cluster utilities are standardized and clipped to $[-1, 1]$ before being used as policy rewards. The LSTM and utility estimator are trained sequentially: gradients from VaDE, direct regression, or KDE fitting are not propagated back into the LSTM. Tables 4 and 5 report the configurations used for the reported experiments.

For the BLUR-KDE ablation, we use a Gaussian kernel. Hidden features are standardized using statistics fitted on the reward-model training set, and conditional utility is estimated by

$$\widehat{U}(\mathbf{h}) = \frac{\sum_i K_\sigma(\mathbf{h}, \mathbf{h}_i) y_i}{\sum_i K_\sigma(\mathbf{h}, \mathbf{h}_i) + \varepsilon}. \tag{39}$$

The bandwidth and reference-set size are kept fixed within each ablation comparison.

Reported SASR configuration. The adapted SASR implementation uses $\lambda_{\mathrm{SASR}} = 0.6$, dense environment-reward weight 0, terminal win/loss weight 1, reward range $[0, 1]$, beta sampling enabled, 1000 random Fourier features, KDE bandwidth 0.2, state-retention rate 0.1, success/failure buffer capacity $200{,}000$, and a minimum of 128 retained samples in each class before nonzero estimated counts are used.

H.8 Randomness and Statistical Reporting

Each experiment is conducted using five independent random seeds controlling network initialization, exploration, replay sampling, minibatch selection, and reward-model initialization. The same set of five seeds is used across compared methods.

After linear interpolation onto each matched task–backbone–comparison global-update grid, learning curves report the mean win or success rate across the five seeds at each aligned grid point. The current figures display the mean curves without uncertainty bands. Main result tables instead summarize late-training performance using the $L = 10$ final-window protocol defined in Section H.4. Their sample standard deviations are computed across the five seed-level window means. Evaluation episodes within one checkpoint are not treated as independent training seeds, and no single best checkpoint is selected for the main results.

All curves are plotted only over the common-support range of their displayed task–backbone–comparison tuple.

Table 4: Reported SMAC configuration.

Setting	Value
QMIX (lr, γ, batch, replay)	$(5\times10^{-4}, 0.99, 32, 5000)$
QMIX target interval; hidden/mixer dims	200; $64/32$
MAPPO actor/critic lr; rollout episodes	$10^{-4}/10^{-4}$; 8
MAPPO clip; epochs; entropy coefficient	0.1; 2; 0.005
Reward LSTM; VaDE latent/clusters	64; $16/8$
Reward-model contextual input	global state
VaDE epochs; batch; lr	30; 256; 10^{-3}
VaDE reconstruction/KL weights	$1.0/0.05$
Auxiliary prediction-loss weight	0.1
Relative window; exploration ratio	100; 0.30
Raw/modeled relative mixture	$0.5/0.5$
Relative fit start/interval/history	$500/500/3000$
Relative gates (label std/reward std/entropy)	$0.1/0.03/0.2$
Bootstrap threshold/internal cap	$0.15/50{,}000$
Absolute fit sample size	512 trajectories
Absolute sampler ratios	$0.20/0.20/0.20/0.15/0.15/0.10$
Absolute gates (sign/std/entropy)	$0.55/[10^{-3}, 10]/0.20$
Terminal win bonuses (raw/absolute)	$1/10$
Success Replay capacity; initial target/attempts	5000; $3000/10{,}000$
Success/near-success replay ratio	$0.8/0.2$
Success-update probability/floor/decay	$0.5/0.1/50{,}000$
Refill threshold/target/attempts	$> 0.1/500/3000$
Final policy initialization; absolute model	independent; frozen
Reported comparison horizon	comparison-specific common support
Absolute-policy internal cap	$500{,}000$
Evaluation interval/episodes	$500/16$

Table 5: Reported HighwayEnv configuration.

Setting	Value
Relative-stage update cap	$50{,}000$
Evaluation interval/episodes	$1000/10$
Maximum episode steps	200
Initial relative trajectories/window	$32/100$
Relative normalization/clipping	disabled/0
Relative-model update interval/epochs	$1/3$
Relative success threshold	0.6
Relative/absolute shaping scales	$0.5/0.5$
Terminal success scale	10
Absolute collection/fit rounds/minimum data	$1/1/32$
Absolute success oversampling ratio	0.5
Success Replay capacity	5000
Reward LSTM; VaDE latent/clusters	128; $16/8$
Reward-model contextual input	state (flattened observation)
Encoder/VaDE epochs	$3/3$
VaDE batch/lr/reconstruction/KL weights	$256/10^{-3}/1/1$
DQN lr/batch/replay/target interval	$5\times10^{-4}/64/10^{5}/1000$
DQN train-start transitions	1000
PPO clip/epochs/entropy coefficient	$0.2/4/0.01$
Final policy initialization; absolute model	independent; frozen
Reported comparison horizon	comparison-specific common support

H.9 t-SNE Visualization Protocol

The RQ4 visualization projects VaDE latent representations on `8m`. The displayed rows encode cluster assignment, episode-level performance, normalized timestep ratio, win/loss outcome, and total policy reward, as described in Figure 6. The three checkpoints are selected post hoc as representative qualitative states spanning Early unconverged behavior with negligible task success, Intermediate behavior with non-trivial success, and late Converged behavior with high task performance. They are used only for visualization and do not affect model selection, final-window evaluation, or any reported quantitative result.

I Experimental Figures

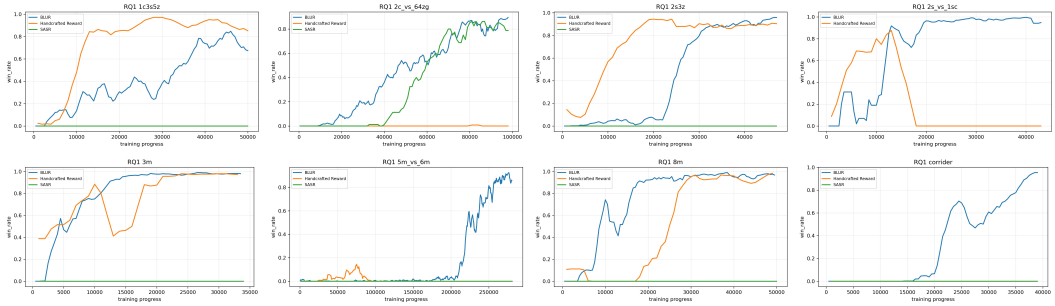

Figure 3: SMAC test win-rate learning curves for RQ1, comparing BLUR, the handcrafted dense-reward reference, and SASR under the QMIX backbone on the eight displayed scenarios. Within each panel, records are linearly interpolated onto the matched QMIX global policy-update grid and truncated to the support shared by all compared methods and seeds; see Table 3 and the main text for the corresponding $L = 10$ final-window summaries. The vertical axis shows test win rate on a 0–1 scale. Curves report the mean across five random seeds at each aligned grid point. No shaded uncertainty region is displayed.

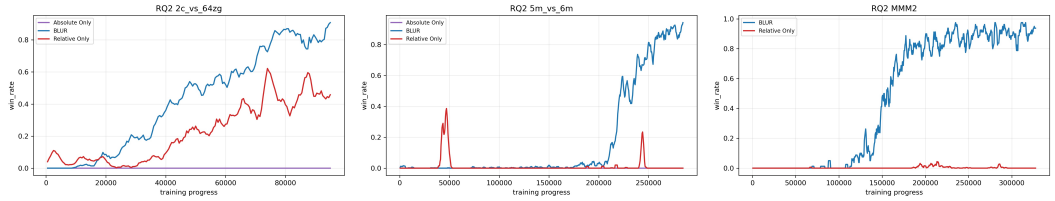

Figure 4: Checkpoint-level stage-ablation learning curves for RQ2 on `2c_vs_64zg`, `5m_vs_6m`, and `MMM2`. BLUR denotes the complete relative-to-absolute training procedure; *Relative Only* omits the transition to a newly trained absolute reward model, and *Absolute Only* removes relative bootstrapping. All variants use QMIX; records are linearly interpolated onto the common-support QMIX global policy-update grid within each displayed comparison. Interaction overhead is detailed in Section F.3. The vertical axis shows test win rate on a 0–1 scale, and curves report the mean across five random seeds. No shaded uncertainty region is displayed.

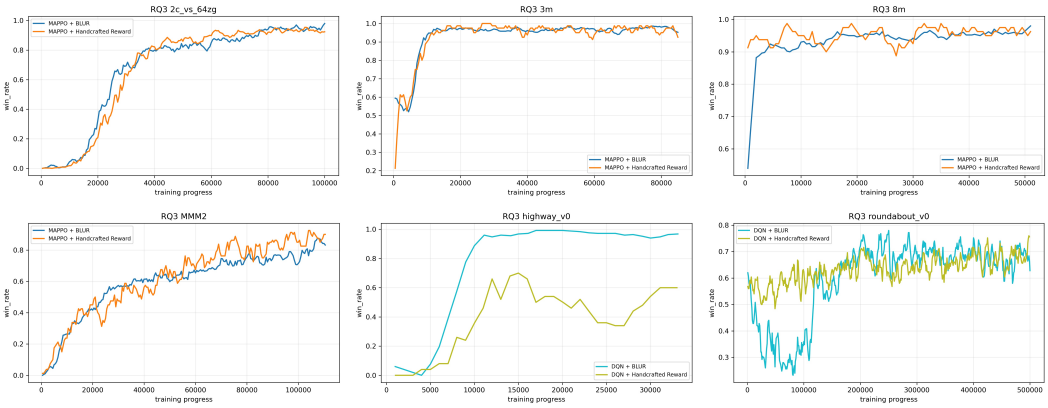

Figure 5: Checkpoint-level policy-backbone comparisons for RQ3. The SMAC panels compare MAPPO with BLUR rewards against MAPPO with handcrafted dense rewards on `2c_vs_64zg`, `3m`, `8m`, and `MMM2`. The HighwayEnv panels compare DQN with BLUR rewards against DQN with handcrafted rewards on `highway-v0` and `roundabout-v0`; see the main text for the PPO final-window results. Records are linearly interpolated onto a common-support global policy-update grid separately within each displayed MAPPO or DQN comparison. The vertical axis shows test win rate for SMAC and test success rate for HighwayEnv, both on a 0–1 scale. Curves report the mean across five random seeds; see the main text for the corresponding final-window summaries over $L = 10$ aligned grid points. No shaded uncertainty region is displayed.

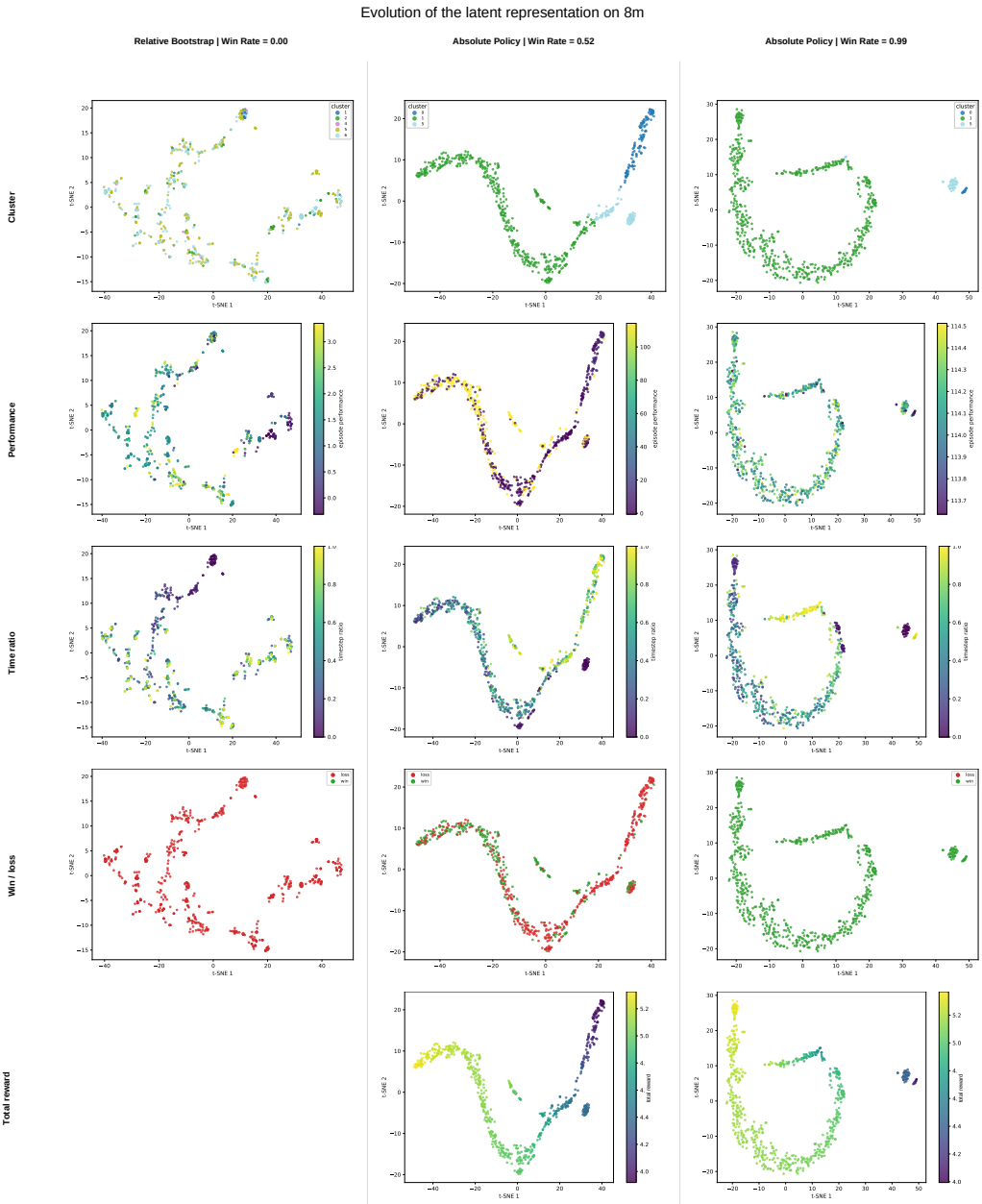

Figure 6: Qualitative t-SNE visualizations of learned trajectory representations at three representative policy states on 8m: Early, an unconverged Relative Utility Bootstrap state; Intermediate, an Absolute Policy Optimization state with non-trivial task success; and Converged, a late Absolute Policy Optimization state with high task performance. The corresponding observed test win rates are 0.00, 0.52, and 0.99, respectively. Rows color the same latent samples by VaDE cluster assignment, episode-level performance, normalized timestep ratio, win/loss outcome, and total policy reward; total-reward panels are shown only for Intermediate and Converged. Colors and markers follow the legends displayed in each row. The visualization is qualitative and is not used for checkpoint selection or quantitative model comparison.

J Discussion and Limitations

Reward-engineering scope. BLUR does not eliminate task-specific prior knowledge. It requires a trajectory-level performance function constructed from task-relevant episode statistics and configured weights. The method uses this coarse evaluation to learn context-dependent dense shaping signals over behavioral patterns, reducing the need to manually specify rewards for individual transitions. The learned shaping rewards are not constrained to sum to $P^{\mathrm{abs}}(\tau)$ and should not be interpreted as causal marginal contributions of individual behaviors.

Timestep-weighted utility aggregation. Each valid timestep contributes one posterior-weighted sample to the cluster utility statistic. Longer trajectories, or trajectories that repeatedly visit the same latent behavioral pattern, can therefore contribute proportionally more weight to that statistic. The reported estimator does not explicitly correct for this visitation-frequency or duration weighting.

Distribution shift under target-policy optimization. The Absolute Reward Model is trained on retained bootstrap and collector trajectories before being frozen and applied to an independently initialized target policy. The evolving target policy may visit regions underrepresented in the reward-model fitting data. Domain-specific stratified or success-oversampled fitting broadens coverage but does not guarantee support for all visited behaviors. The current method does not explicitly estimate reward uncertainty or reject out-of-distribution inputs.

Training overhead and budget interpretation. BLUR introduces reward-model fitting, latent clustering, and additional trajectory collection. Although reported methods are evaluated on common global-update support within each matched task–backbone comparison, this does not match total environment interactions or wall-clock computation. Reward-model fitting, readiness collection, exploration data, and success-replay collection can add interactions without policy updates. The experiments therefore do not establish environment-step sample-efficiency superiority.

Scope of empirical evidence. The evaluation covers cooperative SMAC scenarios and two single-agent HighwayEnv tasks. These results demonstrate compatibility with the tested policy optimizers but do not establish broad domain generalization. The t-SNE analysis is qualitative and may reflect both trajectory-performance association and temporal episode progression; it does not identify a unique or causal utility representation.