SEMANTIC INFORMATION-BOTTLENECK GUIDED MULTI-AGENT COMMUNICATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Multi-agent reinforcement learning relies critically on inter-agent communication for effective coordination in partially observable environments. Recent work has pursued either information-theoretic compression for efficiency or language grounding for interpretability, yet neither addresses the fundamental relationship between what information to communicate and how precisely to communicate it. To address this gap, we introduce **S**emantic **I**nformation-bottleneck **G**uided **M**ulti-**A**gent communication (SIGMA), a framework combining these two perspectives through the variance parameter σ of a variational communication encoder. SIGMA uses σ to quantify uncertainty from partial observability and to weight messages in Bayesian fusion, while a KL regularizer guides the latent space during pretraining; at execution, information flow is governed by confidence-weighted fusion. Concretely, SIGMA leverages a one-time LLM annotation of a small set of real observations to pretrain a semantically structured latent space; agents then transmit distribution parameters from this space conditioned on their local views, and the encoder is jointly optimized with the policy network, allowing σ to weight messages by confidence. Empirically, on standard MARL benchmarks, SIGMA achieves strong performance, remains robust under reduced communication budgets, and transfers effectively to unseen tasks.

1 INTRODUCTION

Multi-agent reinforcement learning (MARL) has become a cornerstone for decentralized decision-making in complex systems, with applications spanning autonomous driving Hua et al. (2025); Antonio & Maria-Dolores (2022), real-time strategy games Li et al. (2025); Wang et al. (2026), and traffic signal control Zhang et al. (2024; 2019). A central challenge in cooperative MARL is partial observability, where each agent perceives only a local view of the global state, rendering purely independent policies insufficient for effective coordination. Inter-agent communication offers a natural solution, enabling agents to share local information and coordinate their actions Ding et al. (2024b).

Despite its promise, recent communication learning work Yuan et al. (2022); Sun et al. (2024) in MARL has followed two largely separate trajectories. One line of work pursues information-theoretic compression through discrete autoencoders or information bottlenecks, trading off message complexity against task utility (Lin et al., 2021; Tucker et al., 2022). These methods achieve high communication efficiency, but the resulting messages are semantically opaque and tightly coupled to the training task, limiting their interpretability and transferability.

Another line of work leverages large language models to ground communication in natural language, enhancing interpretability through semantic alignment (Li et al., 2024; Du et al., 2026). However, these approaches treat the degree of compression as a fixed hyperparameter, determined by codebook size, message dimension, or bottleneck strength, rather than adapting it to the agent's contextual uncertainty. Neither trajectory addresses a foundational question: given that each agent perceives only a partial view of the world, how precisely should it communicate what it observes?

To address this issue, we introduce Semantic Information-bottleneck Guided Multi-Agent communication (SIGMA), a framework that establishes a unified role for the variance parameter σ of a variational communication encoder. In our framework, σ serves as a tractable proxy for the agent's partial-observability uncertainty, with larger σ reflecting lower confidence. At the same time, the KL divergence term in the β-VAE objective serves as an IB-inspired regularizer. By letting the

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.

same parameter serve both purposes, we link how uncertain an agent is to how much its message should influence fusion. Concretely, a confident agent (small σ) sends a message that dominates fusion, while an uncertain agent (large σ) sends a message that is automatically down-weighted, so unreliable information has limited influence on others' decisions. This leads to our core principle: messages should be weighted by the agent's uncertainty.

To instantiate this principle with minimal environment interaction, SIGMA uses a one-time LLM annotation of a small set of real observations to pretrain a semantically structured latent space. Agents then transmit distribution parameters from this space conditioned on their local views, and the encoder is jointly optimized with the policy network, allowing σ to weight messages by confidence. The LLM annotations provide a shared semantic frame across observations, allowing the latent space to capture entity-level semantics that remain meaningful across SMAC maps. These design choices enable uncertainty-weighted communication and coordination, and transfer across scenarios sharing entity-level semantics.

Empirically, we evaluate SIGMA on standard MARL benchmarks (SMAC Samvelyan et al. (2019), LBF Papoudakis et al. (2021), GRF Kurach et al. (2020)). Results show SIGMA outperforms baselines with only a one-time offline LLM annotation, transfers to unseen SMAC maps via a lightweight adapter, and yields interpretable uncertainty estimates correlating with partial observability. Our contributions are: (i) a unified role for the variance parameter σ of a variational communication encoder, simultaneously an uncertainty proxy and a confidence weight for Bayesian fusion, linking confidence to communication precision; (ii) SIGMA, a framework combining LLM-guided semantic pretraining, probabilistic communication, and uncertainty-weighted fusion with dual-speed updates preserving semantic structure; and (iii) validation of SIGMA's performance, transferability, and robustness on MARL benchmarks, with consistent gains over strong baselines.

2 Related Work

Communication Learning in MARL. Communication learning has been extensively studied Ding et al. (2024a); Jiang et al. (2020) in MARL, with methods broadly falling into continuous and discrete paradigms. Early approaches such as CommNet (Sukhbaatar et al., 2016) and IC3Net (Singh et al., 2019) pioneered end-to-end learning of continuous real-valued messages, where agents broadcast vectors optimized via policy gradients. These methods achieve strong task performance but incur high communication bandwidth and produce semantically opaque representations. Subsequent work introduced attention mechanisms (Das et al., 2019) and Transformer architectures (Hu et al., 2024). While these improve coordination performance, they remain fundamentally continuous and deterministic, inheriting the same limitations in interpretability and uncertainty modeling.

To reduce communication overhead, several approaches have explored discretizing messages. aeComm (Lin et al., 2021) employs an autoencoder to compress observations into discrete symbols, achieving high compression rates while grounding symbols through reconstruction. VQ-VIB (Tucker et al., 2022) incorporates the information bottleneck principle to learn a discrete vocabulary, explicitly trading off utility, informativeness, and complexity. These methods achieve high communication efficiency, but the resulting symbols are semantically opaque and tightly coupled to the training task, limiting interpretability and transferability.

LLM-Guided Communication. Recent advances in large language models have inspired new approaches to grounding agent communication in human-understandable semantics. LangGround (Li et al., 2024) uses LLM-generated expert trajectories to align continuous communication vectors with natural language embeddings, enabling human-interpretable communication. GLC (Du et al., 2026) extends this by learning discrete communication protocols from LLM-generated datasets, combining autoencoder-based compression with contrastive alignment for interpretability. LMAC (Bae et al., 2026) uses LLMs to iteratively design rule-based protocols without environment interaction.

While these methods improve interpretability, they treat the degree of compression as a fixed hyperparameter determined by codebook size, message dimension, or bottleneck strength, ignoring contextual uncertainty. Moreover, they rely on deterministic communication and do not model their belief uncertainty, which is critical for robust coordination under partial observability.

Relation to Prior Work. SIGMA differs from all of the above in a fundamental way. Rather than treating compression as a fixed hyperparameter or a static objective, SIGMA establishes a dual role

for the variance parameter σ of a variational communication encoder. In SIGMA, σ serves as a tractable proxy for the agent's partial-observability uncertainty about the global state. At the same time, Bayesian fusion uses σ as a confidence weight to modulate the effective influence of messages. This design directly links two quantities that are usually treated separately, the agent's confidence and its communication precision. Compression-oriented methods keep communication fixed; SIGMA weights it by uncertainty. Interpretability-oriented methods produce deterministic messages; SIGMA grounds them probabilistically with explicit uncertainty quantification. The result is a unified framework that bridges compression efficiency, semantic interpretability, and uncertainty-aware coordination. The IB terminology refers to the KL regularizer used during pretraining; the effective information flow at execution is governed by the confidence-weighted fusion.

3 METHODOLOGY

We now present SIGMA, a framework that learns probabilistic, semantically structured communication protocols for MARL. We first give an overview of the architecture, then detail each component, and finally describe how the communication module integrates with value-based frameworks.

3.1 OVERALL FRAMEWORK

SIGMA is a plug-and-play communication module for centralized training with decentralized execution (CTDE) MARL frameworks. The core idea is to learn a variational encoder that maps each agent's observation to a distribution over latent codes, with the variance parameter σ serving dual roles, quantifying partial-observability uncertainty and providing a confidence weight for Bayesian message fusion. This design enables agents to weight messages by confidence, so that reliable messages dominate fusion while uncertain ones are down-weighted.

Figure 1 illustrates the overall architecture. Each agent maintains a semantic encoder that produces distribution parameters (μ_i, σ_i) from its local observation o_i. The agent transmits these parameters to others, who perform Bayesian fusion to aggregate messages from multiple sources. The fused representation z_f^i is then fed into each agent's local utility network $Q_i(\tau_i, z_f^i, \cdot)$, which also conditions on the agent's history trajectory τ_i. The fused variance σ_{fuse} is used only to compute the fusion weights and is not fed into the utility network; its effect on the policy is indirect through $z_f^i = \mu_{\text{fuse}}$. The local utilities are combined through a monotonic mixing network to produce the joint action-value Q_{tot} for centralized training.

3.2 SEMANTIC VARIATIONAL ENCODER

At the core of SIGMA is a variational encoder that maps each agent's observation to a distribution over latent codes. For agent i with local observation o_i, the encoder produces

$$q_\phi(z_i \mid o_i) = \mathcal{N}(\mu_\phi(o_i), \text{diag}(\sigma_\phi^2(o_i))), \tag{1}$$

where $\mu_\phi(o_i)$ is the mean vector and $\sigma_\phi^2(o_i)$ is the variance vector. The agent transmits the pair (μ_i, σ_i) to other agents as its communication message. The variance parameter σ_i^2 is the cornerstone of SIGMA. First, it quantifies uncertainty from partial observability: when the observation is ambiguous, the encoder outputs a larger σ^2; when clear, a smaller one. Second, it provides a confidence signal for communication. Larger σ indicates a more spread-out posterior and lower confidence; through inverse-variance Bayesian fusion, such messages are down-weighted. The KL term in the β-VAE is an IB-inspired regularizer; the effective information flow is determined by the fusion weights. In the β-VAE objective, the KL divergence term regularizes the posterior:

$$D_{\text{KL}}(q_\phi(z \mid o) \,\|\, p(z)) = \frac{1}{2} \sum_{j=1}^{d} \left(\mu_j^2 + \sigma_j^2 - \log \sigma_j^2 - 1 \right). \tag{2}$$

The KL penalty is minimized at $\sigma^2 = 1$, while the reconstruction loss encourages σ^2 to deviate when more information needs to be encoded. This interplay allows the same σ that reflects uncertainty to also govern the information content. By letting σ serve both roles, we link uncertainty to the effective influence of a message: low-σ messages are trusted and dominate fusion, while high-σ

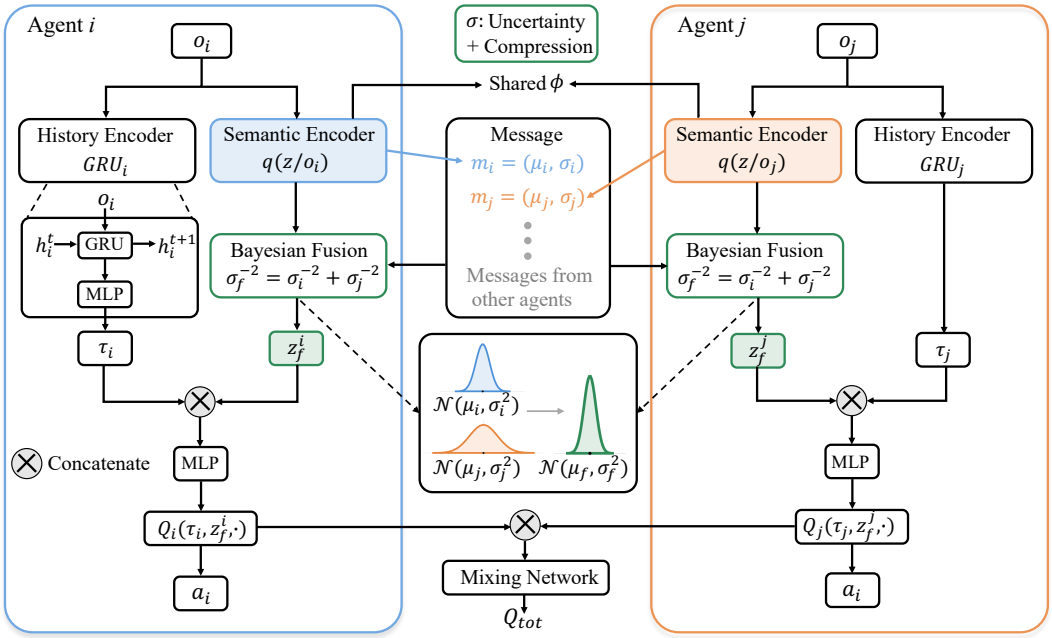


Figure 1: The SIGMA framework. Each agent encodes its local observation into a Gaussian distribution (μ, σ) and exchanges the distribution parameters with other agents. Received messages are fused via inverse-variance weighting, and the fused representation z_f^i, together with the history trajectory τ_i, is fed into the local utility network $Q_i(\tau_i, z_f^i, \cdot)$. The local utilities are combined by a mixing network to produce the joint action-value Q_{tot}. The shared encoder ϕ and the dual role of σ (uncertainty and compression) are highlighted.

messages are down-weighted. Adaptive communication refers to this uncertainty-weighted effective information flow, not to a reduction in nominal message size. This uncertainty-weighted fusion is the key mechanism through which SIGMA achieves data-efficient communication.

3.3 LLM-Guided Semantic Pretraining

To instantiate this principle with minimal environment interaction, SIGMA first constructs a semantically structured latent space from a small set of real observations, using LLM-generated descriptions as semantic anchors.

Data Collection and Semantic Annotation. We run a random or heuristic policy for a limited number of episodes and collect M local observations $\{o^{(n)}\}_{n=1}^{M}$, where $M \approx 1\text{K}{-}5\text{K}$ suffices in all our benchmarks. For each observation $o^{(n)}$, we query an LLM once to generate a natural language description $d^{(n)}$ of the agent's local view, covering the types and relative positions of nearby entities, relevant environmental features, and information useful for coordination. This yields the annotated dataset $\mathcal{D} = \{(o^{(n)}, d^{(n)})\}_{n=1}^{M}$.

VAE Pretraining. We train a β-VAE on paired data $\{(o_i, d_i)\}$, where o_i is the observation vector and d_i its LLM description. Each o_i becomes a textual prompt for LLM, but VAE consumes original o_i and the frozen Sentence-BERT embedding $e(d_i)$. The decoder reconstructs both, minimizing

$$\mathcal{L}_{\text{pretrain}} = -\mathbb{E}_{q_\phi(z|o)}\left[\log p_\theta(o \mid z)\right] + \lambda_s \left(1 - \cos(e(d), \hat{e})\right) + \beta D_{\text{KL}}(q_\phi(z \mid o) \,\|\, p(z)), \quad (3)$$

where $q_\phi(z|o)$ outputs (μ, σ), z is reparameterized, and $\hat{e}$ is the decoded semantic embedding. The observation branch uses MSE for continuous vectors or cross-entropy for discrete ones; the semantic branch uses negative cosine similarity. This updates ϕ, θ while freezing Sentence-BERT. The full architecture and λ_s are detailed in Appendix B.3. Pretraining yields a latent space where similar observations map nearby, and σ^2 tends to reflect observational ambiguity. Empirically, this structure captures shared entity-level semantics across maps, which we leverage for transfer.

3.4 Probabilistic Communication and Bayesian Fusion

During online execution, agents communicate by transmitting their distribution parameters (μ_i, σ_i). We transmit distribution parameters rather than samples, because sampling introduces stochasticity in the policy input and prevents principled Bayesian fusion. This enables deterministic execution and uncertainty-weighted aggregation. Transmitting multiple samples when σ is large is left for future work. The receiving agent aggregates incoming messages using principled Bayesian fusion. We denote the set of communication neighbors of agent i as $\mathcal{C}(i)$. When agent i receives messages from other agents, it updates its belief by combining its own estimate with all incoming messages. Assuming the messages from all agents in $\{i\} \cup \mathcal{C}(i)$ represent independent Gaussian beliefs, the optimal fusion rule is

$$\mu_{\text{fuse}} = \frac{\sum_{k\in\{i\}\cup\mathcal{C}(i)} \sigma_k^{-2}\mu_k}{\sum_{k\in\{i\}\cup\mathcal{C}(i)} \sigma_k^{-2}}, \tag{4}$$

$$\sigma_{\text{fuse}}^{-2} = \sum_{k\in\{i\}\cup\mathcal{C}(i)} \sigma_k^{-2}. \tag{5}$$

The independence assumption is approximate; we therefore report a temperature-scaled version of the fused variance for calibration, which requires only a scalar and adds negligible overhead. This scaling affects only the reported uncertainty and does not alter the fused mean or the policy, since the fused variance is not fed into the utility network. A more principled treatment such as covariance intersection is discussed in Appendix C.3. For the two-agent case $\mathcal{C}(i) = \{j\}$, these expressions simplify to the forms in Figure 1. This rule has an intuitive interpretation: messages with smaller variance (higher confidence) receive greater weight, while uncertain messages are naturally down-weighted. The fusion is fully permutation-invariant and scales naturally to any number of agents.

The fused mean μ_{fuse}, denoted as z_f^i, serves as the communication-enhanced representation for agent i. Along with the agent's history trajectory τ_i, z_f^i is fed into the local utility network $Q_i(\tau_i, z_f^i, \cdot)$. We do not feed σ_{fuse} into the utility network; σ only determines how much each message contributes to z_f^i. The temperature scalar T is learned on a small held-out set with global states available during centralized training by minimizing the NLL of state reconstruction error, and then fixed during execution, so no global state is required at deployment. T is used only to calibrate the reported uncertainty; it does not participate in action selection.

3.5 Joint Optimization with Value-Based CTDE

After pretraining, SIGMA is integrated into a value-based CTDE framework. In our implementation, we adopt QMIX as the base learner, though SIGMA is compatible with other value decomposition methods such as VDN and QPLEX. A critical design choice is the dual-speed update strategy: the utility networks and the mixing network are updated every step, while the VAE is updated slowly to preserve the semantic structure established during pretraining.

Fast Utility and Mixing Updates. At each timestep, agent i's observation o_i is processed by a GRU-based history encoder to produce τ_i, which is then fed into the semantic encoder to obtain (μ_i, σ_i). The agent transmits this message to neighbors, receives messages from them, performs Bayesian fusion (Eqs. 4–5), and feeds the fused representation $z_f^i = \mu_{\text{fuse}}$ (together with τ_i) into the local utility network $Q_i(\tau_i, z_f^i, a_i)$. The local utilities are then combined by a monotonic mixing network to produce the joint action-value:

$$Q_{\text{tot}}(\tau, \{z_f^i\}, \mathbf{a}) = \text{Mix}\left(\{Q_i(\tau_i, z_f^i, a_i)\}_{i=1}^N \mid s\right), \tag{6}$$

where the mixing network takes the global state s as an additional input. The entire system is trained end-to-end using the standard TD loss:

$$\mathcal{L}_{\text{TD}}(\psi) = \mathbb{E}\left[\left(r_t + \gamma \max_{\mathbf{a}'} Q_{\text{tot}}(\tau_{t+1}, \{z_{t+1}^i\}, \mathbf{a}'; s_{t+1}) - Q_{\text{tot}}(\tau_t, \{z_t^i\}, \mathbf{a}_t; s_t)\right)^2\right], \tag{7}$$

where ψ denotes the parameters of the utility networks, the mixing network, and the GRU-based history encoders.

Slow Semantic Updates. The VAE encoder and decoder (ϕ, θ) are updated periodically on observations encountered during policy rollout. Since the LLM is not queried online, the semantic description d is unavailable; we therefore use only the observation reconstruction and KL terms:

$$\mathcal{L}_{\text{sem}} = -\mathbb{E}_{q_\phi(z|o)}[\log p_\theta(o \mid z)] + \beta D_{\text{KL}}(q_\phi(z \mid o) \,\|\, p(z)). \tag{8}$$

The semantic structure is preserved from pretraining. We adopt three strategies to ensure slow updates: (i) the VAE is updated once every K episodes, $K \in [10, 100]$; (ii) its learning rate is an order of magnitude smaller; and (iii) the KL weight β is kept fixed. This design preserves the semantic structure while the utility networks learn to exploit the communication channel.

Total Training Objective. The overall objective combines the TD loss and the semantic consistency loss: $\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{TD}} + \alpha \cdot \mathcal{L}_{\text{sem}}$, where $\mathcal{L}_{\text{TD}}$ updates the utility networks, the mixing network, and the GRU history encoders, while $\mathcal{L}_{\text{sem}}$ updates the VAE parameters ϕ and θ. The TD gradient does not flow into the VAE.

4 Experiments

In this section, we conduct a comprehensive empirical evaluation of SIGMA. We design experiments to answer the following questions:

Q1 (Task Performance): How does SIGMA's performance compare with baseline methods?

Q2 (Communication Efficiency): How does SIGMA's efficiency compare with baseline methods?

Q3 (Uncertainty-Aware Communication): Does σ capture uncertainty from partial observability and drive adaptive communication?

Q4 (Ablation Study): What are the individual contributions of SIGMA's key components?

Q5 (Cross-Task Transfer): How well does SIGMA generalize to unseen tasks?

Q6 (Scalability): How does SIGMA perform as the number of agents increases?

We evaluate SIGMA on 4 benchmarks: SMAC-Comm (Samvelyan et al., 2019), SMACv2 (Ellis et al., 2023), Level-Based Foraging (LBF) (Papoudakis et al., 2021), and Google Research Football (GRF) (Kurach et al., 2020). Baselines include LMAC Bae et al. (2026), CommFormer Hu et al. (2024), MASIA Guan et al. (2022), TarMAC Das et al. (2019), NDQ Wang et al. (2020) and one non-communicating baseline QMIX Rashid et al. (2020). SIGMA is implemented on top of QMIX (Rashid et al., 2020). The VAE encoder uses a 64-dimensional latent space with a Gaussian prior $\mathcal{N}(0, I)$. The semantic descriptions are generated using GPT-5.6 Sol with a one-time offline annotation of 5K samples per environment. All results are averaged over 5 random seeds with standard deviations. **For comprehensive details regarding implementation details, baseline descriptions, hyperparameter settings, and additional experiments, please refer to Appendix.**

Task Performance (Q1). Figure 2 reports the test success rates on SMAC-Comm. SIGMA achieves the highest performance and fastest convergence across all maps, owing to uncertainty-aware communication, where Bayesian fusion of distribution parameters suppresses unreliable information.

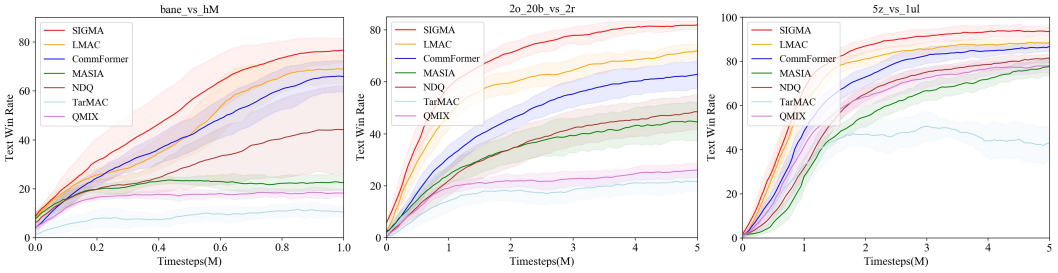


Figure 2: Performance comparison in various SMAC-Comm scenarios.

Table 1 summarizes SMACv2, LBF, and GRF. SIGMA outperforms all baselines on every benchmark, including all SMACv2 races, both LBF settings, and both GRF tasks. Under a fixed data bud-

get, SIGMA matches or exceeds baselines trained with the same budget, and paired t-tests confirm significant improvements over the strongest baseline LMAC ($p < 0.05$; Appendix C.1). TarMAC performs poorly on SMACv2 because its attention mechanism struggles under the high stochasticity and randomized team compositions; we retain its official hyperparameters without per-benchmark tuning, ensuring a fair comparison. TarMAC is included as a representative attention-based baseline; its performance may reflect its original hyperparameters not being tuned for SMACv2. The main comparison is against the strongest baselines, LMAC and CommFormer.

Table 1: Final performance (mean $\pm$ std) on SMACv2, LBF, and GRF. All methods use only local observations at execution.

	SMACv2			LBF		GRF	
Method	terran	protoss	zerg	8x8	11x11	3.vs.1	run_pass
QMIX	62.1 $\pm$ 4.5	47.5 $\pm$ 5.1	34.1 $\pm$ 6.3	84.2 $\pm$ 3.8	77.3 $\pm$ 4.5	69.2 $\pm$ 5.3	67.5 $\pm$ 5.8
TarMAC	31.2 $\pm$ 3.8	22.4 $\pm$ 4.2	16.8 $\pm$ 3.5	79.1 $\pm$ 5.2	70.9 $\pm$ 6.1	72.1 $\pm$ 4.9	70.3 $\pm$ 5.5
NDQ	59.4 $\pm$ 5.9	47.8 $\pm$ 4.7	38.2 $\pm$ 5.4	81.5 $\pm$ 4.0	74.2 $\pm$ 4.8	79.4 $\pm$ 3.7	77.6 $\pm$ 4.2
MASIA	54.3 $\pm$ 7.2	31.9 $\pm$ 6.8	33.5 $\pm$ 8.1	87.4 $\pm$ 3.1	81.0 $\pm$ 3.5	84.1 $\pm$ 3.2	82.3 $\pm$ 3.6
CommFormer	64.8 $\pm$ 3.8	53.7 $\pm$ 4.9	39.2 $\pm$ 5.1	90.5 $\pm$ 2.7	86.1 $\pm$ 3.2	95.1 $\pm$ 1.6	93.8 $\pm$ 1.9
LMAC	68.0 $\pm$ 3.1	57.9 $\pm$ 4.3	41.6 $\pm$ 4.1	93.2 $\pm$ 2.3	89.1 $\pm$ 2.8	96.3 $\pm$ 1.1	97.1 $\pm$ 1.3
SIGMA	**72.1** $\pm$ **2.3**	**61.5** $\pm$ **2.7**	**45.8** $\pm$ **3.0**	**96.0** $\pm$ **0.9**	**92.9** $\pm$ **1.2**	**98.4** $\pm$ **0.6**	**97.7** $\pm$ **0.8**

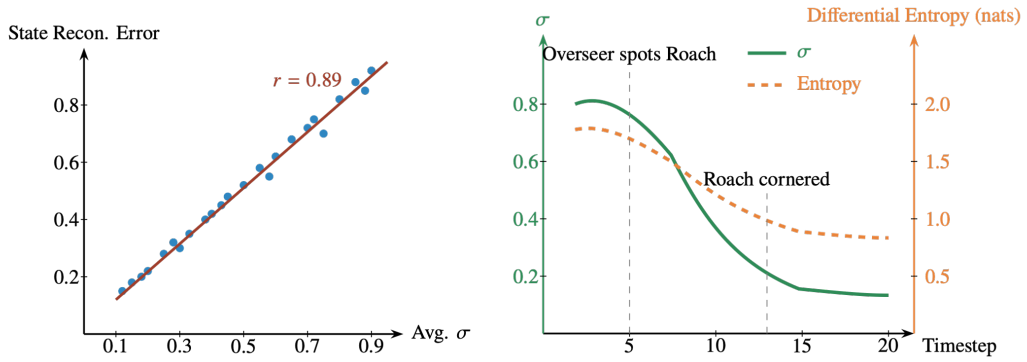
Communication Efficiency (Q2). We evaluate communication efficiency by measuring task performance under varying communication budgets on 2o_20b_vs_2r. Each method is retrained from scratch with different message dimensions, and we report the win rate as a function of the number of transmitted floating-point values per agent per timestep. For SIGMA, the message is (μ_i, σ_i) with latent dimension d, totaling $2d$ values. For continuous baselines, the message dimension is varied over $\{16, 32, 64, 128\}$; discrete methods are retrained under equivalent bit budgets, with details in Appendix C.2. We keep each method's default hyperparameters and do not perform per-budget tuning. Table 2 reports the results. At full budget, SIGMA performs comparably to or slightly better than baselines. As the budget decreases, SIGMA degrades more slowly, because σ allows the receiver to down-weight unreliable messages. SIGMA still transmits $2d$ floats per agent per step; the gain comes from uncertainty-weighted fusion, not from reducing the nominal message size. In contrast, TarMAC's attention weights do not generalize to reduced budgets, and LMAC's fixed protocol cannot adapt its priorities. For each budget K, all methods transmit exactly K floats per agent per step; SIGMA sets $2d = K$. Per-dimension masking guided by σ is left for future work. Robustness to noise/dropout, bit-level normalization, and the sampled-z variant are in Appendix C.2.

Table 2: Win rate (%) under different communication budgets on 2o_20b_vs_2r. Budget denotes the number of floating-point values per agent per step.

Method	**16**	**32**	**64**	**128**
TarMAC	42.1	55.3	65.3	72.8
NDQ	43.5	56.2	66.1	73.5
MASIA	45.8	58.7	68.4	75.2
LMAC	48.2	61.5	71.8	79.4
CommFormer	44.5	57.8	67.1	74.2
SIGMA	**68.4**	**76.2**	**81.9**	**84.5**

Uncertainty-Aware Communication (Q3). We verify σ captures uncertainty from partial observability, the dominant epistemic component in our setting, by measuring its correlation with the agent's ability to recover the global state. Following LMAC's evaluation protocol, we train an auxiliary decoder that infers the global state from the agent's message (μ_i, σ_i) and its local observation history. This decoder is only for evaluation and is not part of SIGMA's training or inference. Per evaluation sample, we record average σ and the state reconstruction error $\|s - \hat{s}\|^2$, and compute the Pearson correlation coefficient between them across approximately 200,000 agent-timestep pairs.

As shown in Figure 3(a), the strong positive correlation confirms that larger σ corresponds to greater difficulty in recovering the global state. After temperature scaling, σ becomes well-calibrated. Controlled occlusion increases σ significantly, while additive observation noise has a smaller effect, indicating that σ primarily encodes uncertainty arising from partial observability. Calibration failures occur mainly under extreme occlusion where σ is overestimated; temperature scaling is preferred over covariance intersection because it is simpler and achieves comparable calibration and performance. Figure 3(b) tracks σ and the differential entropy $H(q_\phi(z|o)) = \frac{1}{2}\log(2\pi e\sigma^2)$ over a trajectory in 2o_20b_vs_2r. The entropy spikes when the Overseers first spot the Roaches and decreases as their locations become predictable, confirming that communication precision adapts to the agent's uncertainty. We disentangle epistemic from aleatoric uncertainty via controlled occlusion and noise; σ primarily tracks partial observability. Under correlated observations, temperature scaling and covariance intersection yield comparable calibration and performance. Theoretical decomposition, calibration details, and additional analyses are in Appendix C.3.



(a) σ vs. state reconstruction error

(b) σ and entropy dynamics

Figure 3: Uncertainty-aware communication (a) σ versus state reconstruction error (state inferred by an auxiliary decoder for evaluation only), confirming that σ reflects uncertainty from partial observability. (b) Dynamics of σ and the differential entropy $H(q_\phi(z|o)) = \frac{1}{2}\log(2\pi e\sigma^2)$ over a trajectory. The entropy measures the effective information carried by the message and follows the same trend as σ.

Ablation Study (Q4). To evaluate the contribution of each component, we conduct an ablation study with 4 variants on 3 SMAC-Comm scenarios (Table 3): w/o LLM (pretraining without semantic annotations), w/o Probabilistic (deterministic encoder transmitting only μ), w/o Bayesian Fusion (simple averaging), and w/o Uncertainty Weighting (fixed σ). Removing LLM guidance or probabilistic communication degrades all 3 maps. Disabling uncertainty weighting causes the most severe drop, confirming that using σ to weight messages is essential. This variant still transmits (μ, σ) but uses fixed fusion weights, so the message content is identical and only the weighting differs. All components are essential, and the full SIGMA achieves the best performance on every map. The w/o Probabilistic variant shares the VAE and semantic pretraining but transmits only μ, isolating the benefit of probabilistic messaging from representation learning; further ablations on LLM variants, fusion, backbones, and hyperparameters, with analyses showing stable performance across smaller LLMs and fewer annotations and negligible annotation cost, are in Appendix C.4.

Table 3: Ablation study on three SMAC-Comm scenarios. Full SIGMA corresponds to the default configuration $d = 64$, matching the budget-128 setting in Table 2.

Variant	**bane_vs_hM**	**2o_20b_vs_2r**	**5z_vs_1ul**
Full SIGMA	76.6 $\pm$4.8	84.5 $\pm$1.6	93.5 $\pm$2.0
w/o LLM	69.7 $\pm$5.1	74.2 $\pm$3.5	85.3 $\pm$2.8
w/o Probabilistic	65.8 $\pm$5.6	70.5 $\pm$4.0	82.2 $\pm$3.2
w/o Bayesian Fusion	71.6 $\pm$4.9	76.3 $\pm$3.3	88.6 $\pm$2.5
w/o Uncertainty Weighting	61.5 $\pm$6.3	65.7 $\pm$4.5	77.7 $\pm$3.6

Cross-Task Transfer (Q5). We evaluate transfer from 2o_20b_vs_2r to unseen SMAC maps. Because these maps have different observation dimensions, we introduce a lightweight adapter that projects target-map observations to the source dimension. The adapter is a linear layer trained with the same reconstruction loss as VAE pretraining on 100 target-map episodes, while the VAE and communication protocol remain frozen. We train SIGMA on 2o_20b_vs_2r, then train the adapter and new utility/mixing networks on bane_vs_hM and 5z_vs_1ul. Results are in Table 4. SIGMA retains most of its source-task performance on both target maps, substantially outperforming LMAC and TarMAC. The adapter only aligns input dimensions and does not learn new semantics; no task-specific rewards or matchup information are used. This transfer is enabled by the shared semantic structure that keeps σ meaningful, so the communication protocol remains valid. Thus, SIGMA achieves effective transfer with only a lightweight adapter. All methods use the same 100-episode target-domain budget. SIGMA freezes the pretrained VAE and communication protocol and trains only the adapter and policy; baselines are trained from scratch following their original designs. SIGMA's pretraining is offline on the source task, with no target-domain data. From-scratch results are in Table 21. The large gap to baselines on target maps reflects that LMAC's protocol is designed for the source task and cannot be re-adapted within 100 episodes, whereas SIGMA's frozen encoder provides a task-agnostic semantic interface. Adapter capacity is analyzed in Appendix C.5.

Table 4: Transfer performance (win rate %) from 2o_20b_vs_2r to unseen SMAC maps. All methods use 100 target-map episodes. SIGMA freezes the VAE and communication protocol, training only the adapter and policy; baselines train from scratch under the same budget.

Target Map	TarMAC	CommFormer	LMAC	MASIA	NDQ	SIGMA
bane_vs_hM	18.7	26.1	32.4	24.5	21.8	**74.8**
5z_vs_1ul	31.2	37.5	43.6	35.7	33.4	**81.0**

Scalability (Q6). We evaluate SIGMA with increasing agent numbers on SMACv2, from 5 vs 5 to 20 vs 20, across three races. Table 5 summarizes win rates. SIGMA consistently outperforms LMAC across team sizes, while TarMAC degrades most noticeably. Per-agent overhead is constant and Bayesian fusion is permutation-invariant. Total bandwidth scales with the number of communication links, and SIGMA's steps to 50% win rate grow more slowly than baselines; detailed results and analyses are in Appendix C.6. At 20 vs 20, all methods degrade somewhat due to harder credit assignment; SIGMA's communication module itself adds no new parameters as team size grows, but overall performance is still bounded by the policy learner. Results are relative comparisons under identical training budgets and evaluation protocols, with official hyperparameters for all methods.

Table 5: Scalability on SMACv2 across races and team sizes. Win rates (%), averaged over 5 seeds. Results are relative comparisons under identical training budgets and evaluation protocols.

Method	terran			protoss			zerg		
	5 vs 5	10 vs 10	20 vs 20	5 vs 5	10 vs 10	20 vs 20	5 vs 5	10 vs 10	20 vs 20
TarMAC	31.2	24.3	18.1	22.4	17.5	13.0	16.8	13.1	9.7
LMAC	68.0	59.2	47.8	57.9	50.7	40.2	41.6	36.0	29.5
SIGMA	**72.1**	**62.3**	**51.7**	**61.5**	**53.6**	**44.9**	**45.8**	**37.5**	**32.0**

5 Conclusion and Future Work

We introduced SIGMA, which uses the variance parameter σ of a variational communication encoder to estimate partial-observability uncertainty and weight messages by confidence. Requiring only a one-time offline LLM annotation, SIGMA outperforms prior communication methods in task performance, communication efficiency, and transfer across MARL benchmarks. A limitation is its reliance on entity-level semantics shared across maps; extending it to raw pixels or fundamentally different entity types would require a learned semantic extractor, which we leave for future work. Other directions include dynamically adapting the KL weight β based on uncertainty to improve the compression-informativeness trade-off, and leveraging the semantic grounding and uncertainty quantification of SIGMA for human-agent collaboration.

6 Ethics Statement

We promise that we have read the ICLR Code of Ethics, and this article has not raised any questions regarding the Code of Ethics.

7 Reproducibility Statement

We promise that SIGMA is reproducible. We provide the code of our framework and baselines in the supplementary file, including the implementations of the MARL and LLM agents, training scripts, and the environments. Hyperparameters are listed in Appendix B.3, and the exact LLM endpoint and prompts are in Appendix A.4. All random seeds and evaluation protocols are included in the released code.

References

Guillen-Perez Antonio and Cano Maria-Dolores. Multi-agent deep reinforcement learning to manage connected autonomous vehicles at tomorrow's intersections. *IEEE Transactions on Vehicular Technology (TVT)*, 2022.

Sangjun Bae, Yisak Park, Sanghyeon Lee, and Seungyul Han. Llm-guided communication for cooperative multi-agent reinforcement learning. In *International Conference on Machine Learning (ICML)*, 2026.

Abhishek Das, Théophile Gervet, Joshua Romoff, Dhruv Batra, Devi Parikh, Mike Rabbat, and Joelle Pineau. Tarmac: Targeted multi-agent communication. In *International Conference on Machine Learning (ICML)*, 2019.

Gang Ding, Zeyuan Liu, Zhirui Fang, Kefan Su, Liwen Zhu, and Zongqing Lu. Multi-agent coordination via multi-level communication. *Advances in Neural Information Processing Systems (NeurIPS)*, 2024a.

Shifei Ding, Wei Du, Ling Ding, Jian Zhang, Lili Guo, and Bo An. Robust multi-agent communication with graph information bottleneck optimization. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2024b.

Wei Du, Benyu Wu, Yuqing Sun, Wei Guo, Yuntao Du, Zhongmin Yan, Guoxian Yu, and Lizhen Cui. Learning efficient and interpretable multi-agent communication. In *International Conference on Learning Representations (ICLR)*, 2026.

Benjamin Ellis, Jonathan Cook, Skander Moalla, Mikayel Samvelyan, Mingfei Sun, Anuj Mahajan, Jakob Foerster, and Shimon Whiteson. Smacv2: An improved benchmark for cooperative multi-agent reinforcement learning. *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Cong Guan, Feng Chen, Lei Yuan, Chenghe Wang, Hao Yin, Zongzhang Zhang, and Yang Yu. Efficient multi-agent communication via self-supervised information aggregation. *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Shengchao Hu, Li Shen, Ya Zhang, and Dacheng Tao. Learning multi-agent communication from graph modeling perspective. In *International Conference on Learning Representations (ICLR)*, 2024.

Min Hua, Xinda Qi, Dong Chen, Kun Jiang, Zemin Eitan Liu, Hongyu Sun, Quan Zhou, and Hongming Xu. Multi-agent reinforcement learning for connected and automated vehicles control: Recent advancements and future prospects. *IEEE Transactions on Automation Science and Engineering (T-ASE)*, 2025.

Jiechuan Jiang, Chen Dun, and Tiejun Huang. Graph convolutional reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2020.

Karol Kurach, Anton Raichuk, Piotr Stańczyk, Michał Zajac, Olivier Bachem, Lasse Espeholt, Carlos Riquelme, Damien Vincent, Marcin Michalski, Olivier Bousquet, et al. Google research football: A novel reinforcement learning environment. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2020.

Huao Li, Hossein Nourkhiz Mahjoub, Behdad Chalaki, Vaishnav Tadiparthi, Kwonjoon Lee, Ehsan Moradi Pari, Charles Lewis, and Katia Sycara. Language grounded multi-agent reinforcement learning with human-interpretable communication. *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Lihe Li, Lei Yuan, Pengsen Liu, Tao Jiang, and Yang Yu. Llm-assisted semantically diverse teammate generation for efficient multi-agent coordination. In *International Conference on Machine Learning (ICML)*, 2025.

Toru Lin, Jacob Huh, Christopher Stauffer, Ser Nam Lim, and Phillip Isola. Learning to ground multi-agent communication with autoencoders. *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

Georgios Papoudakis, Filippos Christianos, Lukas Schäfer, and Stefano V Albrecht. Benchmarking multi-agent deep reinforcement learning algorithms in cooperative tasks. *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

Tabish Rashid, Mikayel Samvelyan, Christian Schroeder De Witt, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. Monotonic value function factorisation for deep multi-agent reinforcement learning. *Journal of Machine Learning Research (JMLR)*, 2020.

Mikayel Samvelyan, Tabish Rashid, Christian Schroeder de Witt, Gregory Farquhar, Nantas Nardelli, Tim GJ Rudner, Chia-Man Hung, Philip HS Torr, Jakob Foerster, and Shimon Whiteson. The starcraft multi-agent challenge. In *International Conference on Autonomous Agents and MultiAgent Systems (AAMAS)*, 2019.

Amanpreet Singh, Tushar Jain, and Sainbayar. Sukhbaatar. Learning when to communicate at scale in multiagent cooperative and competitive tasks. In *International Conference on Learning Representations (ICLR)*, 2019.

Sainbayar Sukhbaatar, Arthur Szlam, and Rob. Fergus. Learning multiagent communication with backpropagation. *Advances in Neural Information Processing Systems (NeurIPS)*, 2016.

Chuxiong Sun, Zehua Zang, Jiabao Li, Jiangmeng Li, Xiao Xu, Rui Wang, and Changwen Zheng. T2mac: Targeted and trusted multi-agent communication through selective engagement and evidence-driven integration. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2024.

Mycal Tucker, Roger Levy, Julie A Shah, and Noga Zaslavsky. Trading off utility, informativeness, and complexity in emergent communication. *Advances in neural information processing systems (NeurIPS)*, 2022.

Tonghan Wang, Jianhao Wang, Chongyi Zheng, and Chongjie Zhang. Learning nearly decomposable value functions via communication minimization. In *International Conference on Learning Representations (ICLR)*, 2020.

Ziyan Wang, Meng Fang, Tristan Tomilin, Fei Fang, and Yali Du. Safe multi-agent reinforcement learning with natural language constraints. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2026.

Lei Yuan, Jianhao Wang, Fuxiang Zhang, Chenghe Wang, Zongzhang Zhang, Yang Yu, and Chongjie Zhang. Multi-agent incentive communication via decentralized teammate modeling. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2022.

Huichu Zhang, Siyuan Feng, Chang Liu, Yaoyao Ding, Yichen Zhu, Zihan Zhou, Weinan Zhang, Yong Yu, Haiming Jin, and Zhenhui Li. Cityflow: A multi-agent reinforcement learning environment for large scale city traffic scenario. In *The World Wide Web Conference (WWW)*, 2019.

Yutian Zhang, Guohong Zheng, Zhiyuan Liu, Quan Li, and Zeng Haipeng. Marlens: understanding multi-agent reinforcement learning for traffic signal control via visual analytics. *IEEE Transactions on Visualization and Computer Graphics (TVCG)*, 2024.

A Implementation Details

A.1 Training Procedure

Algorithm 1 summarizes the training procedure. SIGMA first pretrains a β-VAE on a small set of real observations, collected via a random or heuristic policy and annotated once by an LLM, using semantic reconstruction and structure-preserving objectives. In the online phase, the VAE is integrated with a value-based CTDE learner, where the utility and mixing networks are updated every step via the TD loss, while the VAE is updated slowly with a semantic consistency loss. This dual-speed update is critical. Updating the VAE at the same rate would let task-specific gradients overwrite the pretrained semantic structure and lose generalizability, whereas slow updates (every K episodes, with an order-of-magnitude smaller learning rate) allow the policy to exploit the communication channel while the encoder remains anchored to the semantically grounded latent space.

Algorithm 1 Training Procedure of SIGMA

Input: Number of pretraining samples N, VAE latent dimension d, slow-update interval K, semantic loss weight α, learning rates $\eta_{\text{VAE}}, \eta_{\text{policy}}$
Initialize: VAE encoder ϕ, decoder θ; utility networks Q_i, mixing network ψ; replay buffer $\mathcal{D}$
Phase 1: LLM-Guided Semantic Pretraining
for $n = 1$ to N **do**
  Collect observation o_n via random or heuristic policy
  Query LLM to obtain semantic description $d_n = \text{LLM}(p, o_n)$
end for
for each pretraining iteration **do**
  Sample minibatch $\{(o, d)\}$ from $\mathcal{D}_{\text{pretrain}}$
  Update (ϕ, θ) by minimizing $\mathcal{L}_{\text{pretrain}} = -\mathbb{E}_{q_\phi}[\log p_\theta(o \mid z)] + \lambda_s(1 - \cos(e(d), \hat{e})) + \beta D_{\text{KL}}$
end for
Phase 2: Online Joint Optimization
for each training iteration **do**
  for each environment step t **do**
    Encode o_i^t through GRU to obtain τ_i^t
    Compute $(\mu_i, \sigma_i) = q_\phi(z \mid o_i^t)$
    Transmit (μ_i, σ_i); perform Bayesian fusion (Eqs. 4–5) to obtain z_f^i
    Select action $a_i = \arg\max_a Q_i(\tau_i, z_f^i, a)$
    Store transition in $\mathcal{D}$
  end for
  Update utility and mixing networks by minimizing $\mathcal{L}_{\text{TD}}$
  if episode $\bmod K = 0$ **then**
    Update (ϕ, θ) with learning rate $\eta_{\text{VAE}} \ll \eta_{\text{policy}}$ by minimizing $\mathcal{L}_{\text{sem}}$
  end if
end for

A.2 Network Architecture

Semantic Encoder. The encoder $q_\phi(z \mid o)$ is a 3-layer MLP with hidden dimensions $[128, 128, 2d]$, outputting both μ and $\log \sigma^2$. We parameterize $\log \sigma^2$ instead of σ^2 for numerical stability and to avoid positivity constraints. The decoder $p_\theta(o, d \mid z)$ mirrors the encoder with hidden dimensions $[128, 128, d_o + d_d]$, where d_o and d_d are the observation and semantic embedding dimensions. The latent space is 64-dimensional with a standard Gaussian prior $\mathcal{N}(0, I)$.

History Encoder. Each agent's observation history is encoded by a GRU with a 128-dimensional hidden state, following standard QMIX. The GRU output τ_i serves as the history representation and is fed into the local utility network together with the fused communication representation z_f^i.

Utility and Mixing Networks. The local utility network Q_i is a 2-layer MLP with 128 hidden units, taking the concatenation of (τ_i, z_f^i, a_i) as input. The mixing network follows the standard QMIX design with a hypernetwork conditioned on the global state, producing the joint action-value Q_{tot} trained via TD learning.

A.3 Training Details

For VAE pretraining we use Adam with learning rate 10^{-3}, batch size 128, and 1000 epochs. Online, the policy uses Adam at 5×10^{-4}, while the VAE uses 10^{-5} and is updated every $K = 50$ episodes; the semantic loss weight is $\alpha = 0.5$ and the KL weight is fixed at $\beta = 0.01$. Learning rates are constant without scheduling. We apply gradient clipping with max norm 10 and Xavier uniform initialization, with no weight decay. The replay buffer stores the most recent 5000 episodes, sampled uniformly. All experiments run on NVIDIA RTX 4090 GPUs.

A.4 LLM Annotation

In SIGMA, the LLM is used exclusively for a one-time offline semantic annotation of a small set of real observations. It does not participate in online training, policy optimization, or inference.

Procedure. We collect a small set of observations $\{o_i\}_{i=1}^{N}$ (typically $N = 5$K) using a random or heuristic policy. For each observation o_i, we query an LLM once to obtain a natural language description $d_i = \mathrm{LLM}(p, o_i)$, where p is a prompt that includes the task description and the semantics of each observation dimension. The resulting pairs $\{(o_i, d_i)\}$ are used to pretrain the VAE with a semantic reconstruction loss. The LLM is never queried again during online training or execution.

Model and Parameters. We use GPT-5.6 Sol with temperature 0.6 and a maximum output length of 128 tokens. The temperature is set to a moderate value to balance diversity and consistency; a lower temperature tends to produce repetitive descriptions, while a higher temperature introduces unnecessary variability. The same model and parameters are used across all environments. All LLM calls were made through the API; we recorded the model snapshot and call dates. The exact prompts, annotated data, and random seeds will be released with the code.

General Prompt Template. The prompt consists of three parts: task description, state and observation description, and output format instruction. Placeholders are filled according to the specific environment.

General Prompt Template

1. Task Description.
{task_description}

2. State Information.
{state_description}

3. Observation Information.
{observation_description}

4. Annotation Instruction.
You are an expert in multi-agent coordination. Given an agent's local observation in the above task, describe in concise natural language what the agent perceives. Your description should include:

- The agent's own status (e.g., position, health, role).
- Visible entities: their types, relative positions, and any relevant attributes (e.g., health, threat level).
- Any contextual information that would be useful for coordination, such as whether a teammate is in danger or an enemy is vulnerable.

Do not mention any information that is not directly observable from the given observation. Do not speculate about the global state beyond what can be inferred from the local view.

5. Observation.
{observation}

6. Output Format.
Return a JSON object with a single key "description" containing your natural language description.

Example: SMAC-Comm 2o_20b_vs_2r. To illustrate how the template is instantiated, we show the complete prompt for the SMAC-Comm 2o_20b_vs_2r scenario. In this scenario, a team of 22 agents (2 Overseers and 20 Banelings) must eliminate two enemy Roaches. The Overseers can directly observe the Roaches, while the Banelings cannot see them and must rely on communication to locate them.

Prompt for SMAC-Comm 2o_20b_vs_2r

1. Task Description.
In this scenario, a team of 22 agents (2 Overseers and 20 Banelings) must eliminate two enemy Roaches. The Overseers can directly observe the Roaches, while the Banelings cannot see them and must rely on communication to locate them. The episode ends when both Roaches are eliminated or all Banelings are dead.

2. State Information.
The global state includes the absolute positions (x, y) of all units, their health, shield, unit type, weapon cooldown, and last action.

3. Observation Information.
Each agent receives a local observation vector of 171 dimensions, which includes:

- Dims 0–3: movement availability (four directions).
- Dims 4–11: nearest visible enemy features (visibility, distance, relative x, relative y, health, shield, unit type, cooldown).
- Dims 12–91: visible ally features (up to 10 allies, each with 8 dimensions covering visibility, distance, relative position, health, shield, and unit type).
- Dims 92–94: self features (health, shield, unit type).
- Dims 95–101: last action one-hot encoding (7 actions).
- Dims 102–123: agent ID one-hot encoding (22 agents).
- Dims 124–170: terrain and contextual features (height, walkability, and other environmental information).

4. Annotation Instruction.
You are an expert in multi-agent coordination. Given an agent's local observation in the above task, describe in concise natural language what the agent perceives. Your description should include:

- The agent's own status (e.g., position, health, role).
- Visible entities: their types, relative positions, and any relevant attributes (e.g., health, threat level).
- Any contextual information that would be useful for coordination, such as whether a teammate is in danger or an enemy is vulnerable.

Do not mention any information that is not directly observable from the given observation. Do not speculate about the global state beyond what can be inferred from the local view.

5. Observation.
{observation}

6. Output Format.
Return a JSON object with a single key "description" containing your natural language description.

Example Output.
For an observation from an Overseer where it sees two Roaches at relative positions (3, 2) and (5, -1), and two Banelings at (1, 1) and (2, -1):
{"description": "I see two enemy Roaches to my northeast, about three and five units away. Two allied Banelings are to my northwest and southeast, both close by and at full health."}

For other environments, the task description, state information, and observation information follow the same structure and are adapted from the corresponding benchmark documentation. The concrete prompt instances for all environments are included in the released code.

Annotation Examples. Table 6 shows representative examples from SMAC-Comm and LBF. The descriptions are concise and focus on coordination-relevant information.

Table 6: Examples of LLM annotations.

Observation (abbreviated)	LLM Description
Overseer sees Roach at relative (3,2), two Banelings nearby at (1,1) and (2,-1).	"I see the enemy Roach to my northeast, about three units away. Two allied Banelings are to my northwest and southeast, both close by."
Baneling sees no enemy, two allies at (1,0) and (-1,1).	"No enemy visible. Two allies are nearby, one to my east and one to my northwest."
LBF agent sees food requiring level 2, one ally at (1,0).	"I see a food item that requires level 2. One ally is to my east. We may need to cooperate to collect it."

Annotation Quality. Although the LLM descriptions are not perfect, they serve as soft semantic anchors rather than hard supervision. The VAE pretraining objective combines a hard observation reconstruction loss with a soft semantic reconstruction and structure loss. The observation reconstruction ensures that the latent code remains grounded in the actual observation, while the semantic loss shapes the geometry of the latent space. Consequently, occasional inaccuracies in the LLM descriptions do not corrupt the latent space; they merely add noise to the semantic prior.

We further verify the effectiveness of the annotations through three mechanisms. First, we manually inspect a random subset of 200 annotations (about 4% of the total) and find that over 90% are factually correct and contain no misleading information. Second, the ablation study (Table 3) shows that removing LLM guidance (w/o LLM) significantly degrades both task performance and transfer, confirming that the annotations provide useful signal. Third, the uncertainty parameter σ remains reliable even if some annotations are noisy, because σ is primarily determined by the observation reconstruction loss and the KL term, not by the semantic loss. This is confirmed by the strong correlation between σ and state reconstruction error ($r = 0.89$) in Figure 3(a).

Cost and Token Consumption. The annotation is performed once per environment. For 5K samples, the total token consumption is approximately 60K input tokens and 12K output tokens, corresponding to a one-time API cost of roughly $0.25 per environment. This cost is negligible compared to the training cost of MARL baselines. No LLM calls are made during online training or inference.

Data collection script. The observation collection uses a random policy for 100 episodes per environment. The LLM is queried once per observation with temperature 0.6. The resulting descriptions are stored as JSON files. The VAE pretraining script reads these files and trains for 1000 epochs. All scripts will be released upon publication.

B Experimental Details

This section provides comprehensive details of our experimental setup, including environment configurations, baseline implementations, and hyperparameter settings.

B.1 Environment Details

We evaluate SIGMA on 4 benchmarks: SMAC-Comm, SMACv2, LBF, and GRF. Each benchmark stresses a different aspect of communication: SMAC-Comm requires precise coordination under partial observability, SMACv2 tests robustness to stochasticity, LBF requires efficient information sharing in a grid world, and GRF involves high-dimensional observations with sparse rewards.

SMAC-Comm (Bae et al., 2026): a communication-intensive variant of SMAC (Samvelyan et al., 2019). We evaluate on three scenarios: bane_vs_hM, 2o_20b_vs_2r, and 5z_vs_1ul. It follows the original SMAC environment: observations are restricted to local information within a fixed 9-unit sight range; the action space consists of discrete movement, attack, and special ability commands; and the reward function includes damage dealt to enemies, enemy eliminations, and a win bonus.

The environment itself provides no communication channel; the "Comm" denotes that these scenarios are communication-intensive, so agents must exchange information through an algorithm-defined protocol to succeed. Table 7 summarizes the configuration of each scenario.

Table 7: SMAC-Comm scenario configurations.

Scenario	Ally Units	Enemy Units	State Dim	Obs Dim
bane_vs_hM	3 Banelings	1 Hydralisk, 1 Medivac	52	31
2o_20b_vs_2r	2 Overseers, 20 Banelings	2 Roaches	296	171
5z_vs_1ul	5 Zealots	1 Ultralisk	63	36

The 2o_20b_vs_2r scenario is particularly communication-critical: only the Overseers can observe the Roaches, and the Banelings must rely entirely on communication to locate them. This scenario also tests scalability, as it involves 22 agents.

SMACv2 (Ellis et al., 2023): a more stochastic version of SMAC that randomizes unit types, positions, and team compositions across episodes. We evaluate on terran_5_vs_5, protoss_5_vs_5, and zerg_5_vs_5. The randomization introduces severe distribution shifts that prevent memorization of fixed scenarios, forcing the policy to learn generalizable coordination strategies rather than map-specific tricks. Table 8 summarizes the configuration of each scenario.

Table 8: SMACv2 scenario configurations. Probabilities specify the sampling distribution over randomized compositions.

Scenario	Ally Units (prob.)	State Dim	Obs Dim
terran_5_vs_5	Marine (0.45), Marauder (0.45), Medivac (0.1)	120	82
protoss_5_vs_5	Zealot (0.45), Stalker (0.45), Colossus (0.1)	130	92
zerg_5_vs_5	Zergling (0.45), Hydralisk (0.45), Baneling (0.1)	120	82

LBF (Papoudakis et al., 2021): a cooperative foraging task on a grid. We use 8x8-2p-2f-s1-coop and 11x11-6p-4f-s1-coop, where $n \times n$ denotes the grid size, p the number of agents, f the number of food items, and s the sight range. Each agent observes only a 3×3 local field centered on itself. Food items require the joint effort of multiple agents, forcing coordination through communication. The larger 11x11 setting introduces more agents and food items, testing scalability. Table 9 summarizes the configuration of each scenario.

Table 9: LBF scenario configurations.

Scenario	Grid	Agents	Food	Sight	Obs Dim
8x8-2p-2f-s1-coop	8 x 8	2	2	1	3 x 3 local
11x11-6p-4f-s1-coop	11 x 11	6	4	1	3 x 3 local

GRF (Kurach et al., 2020): a physics-based soccer simulator. We evaluate on academy_3_vs_1_with_keeper and academy_run_pass_and_shoot. Each agent controls one player and must cooperate with teammates to score against scripted opponents. GRF features high-dimensional observations and sparse rewards, making communication essential for coordination. Table 10 summarizes the configuration of each scenario.

Table 10: GRF scenario configurations.

Scenario	Ally	Opponent	State Dim	Obs Dim
3_vs_1_with_keeper	3 central midfield	1 goalkeeper, 1 center back	26	26
run_pass_and_shoot	2 central back	1 goalkeeper, 1 center back	22	22

B.2 Baseline Descriptions

We compare against QMIX (no communication), TarMAC and NDQ (continuous and discrete differentiable communication), MASIA (sparsity-regularized communication), CommFormer (Transformer-based communication), and LMAC (LLM-designed communication protocols). These baselines cover the main communication paradigms and include the strongest prior method (LMAC), ensuring a comprehensive comparison. All baselines use publicly available implementations and are retrained under identical network capacities, communication budgets, and evaluation protocols.

QMIX (Rashid et al., 2020): A value-decomposition method without explicit communication, serving as a lower bound. It factorizes the joint action-value into individual utilities through a monotonic mixing network. We use the official implementation with standard hyperparameters: Adam optimizer with learning rate 5×10^{-4}, batch size 32, replay buffer of 5000 episodes, target network update interval 200 steps, discount factor $\gamma = 0.99$, and ϵ-greedy exploration annealed linearly from 1.0 to 0.05.

TarMAC (Das et al., 2019): An attention-based method that enables targeted message weighting. Agents broadcast a signature key and value, and receivers compute attention weights via dot-product matching. While effective at selecting relevant messages, TarMAC does not model uncertainty and relies on continuous messages. We use the official implementation with key/query dimension 64 and softmax temperature 1.0. TarMAC is included as a representative attention-based baseline; its performance may reflect its original hyperparameters not being tuned for SMACv2.

NDQ (Wang et al., 2020): A method that minimizes communication overhead through nearly decomposable Q-functions, using information-theoretic regularizers to balance mutual information and entropy. NDQ achieves high communication efficiency but does not adapt the message content to the agent's uncertainty. We use the official implementation with default hyperparameters.

MASIA (Guan et al., 2022): A self-supervised information aggregation method that compresses raw messages into compact, permutation-invariant representations. MASIA improves efficiency but does not explicitly model uncertainty or provide a principled compression mechanism. We use the official implementation with the recommended self-supervised objectives.

CommFormer (Hu et al., 2024): A Transformer-based communication method that learns a sparse communication graph through bi-level optimization. CommFormer uses attention to weight messages but relies on continuous, deterministic messages and does not model uncertainty. We reproduce CommFormer under the QMIX framework with default hyperparameters (2 attention heads, 2 communication rounds, message dimension 64).

LMAC (Bae et al., 2026): An LLM-guided method that uses iterative refinement to design state-aware communication protocols. LMAC never invokes the LLM online; protocols are refined offline using criteria from RL transition data. LMAC is the most relevant baseline because it also uses LLM priors for communication, but its protocols are deterministic and do not model uncertainty.

All baselines are reproduced under the same QMIX framework with their recommended hyperparameters for fair comparison.

B.3 Hyperparameter Setup

We determine the reconstruction threshold α for the auxiliary decoder by aligning the statistical distribution of errors with the physical spatial semantics of each environment. This threshold is used only for the uncertainty–error correlation analysis and does not affect SIGMA's training or inference. For SIGMA, the VAE encoder uses a 64-dimensional latent space with a standard Gaussian prior. The semantic descriptions are generated using GPT-5.6 Sol with a one-time offline annotation of 5K samples per environment. For the semantic branch, we use a frozen all-mpnet-base-v2 Sentence-BERT model, which outputs 768-dimensional embeddings. Freezing the semantic encoder prevents task-specific gradients from corrupting the pretrained semantic space, following the same rationale as freezing the VAE during transfer (Appendix C.5). The semantic loss weight is $\lambda_s = 0.1$. All hyperparameters are fixed across environments and reused unchanged in the transfer and scalability experiments; the VAE is kept frozen during transfer (Appendix C.5). Sensitivity to the latent dimension, KL weight, slow-update interval, and semantic loss weight is analyzed in Appendix C.4. Table 11 summarizes the key hyperparameters.

Table 11: Hyperparameters of SIGMA.

Parameter	Value	Note
Latent dimension d	64	
VAE learning rate (pretrain)	10^{-3}	Adam
KL weight β_1	0.01	
Structure weight λ_s	0.1	
Pretraining samples N	5K	per environment
Batch size	128	
Training epochs	1000	
VAE learning rate (online)	10^{-5}	
Policy learning rate	5×10^{-4}	
Slow-update interval K	50	episodes
Semantic loss weight α	0.5	
Discount factor γ	0.99	
GAE parameter λ	0.95	
Clip range	0.2	
Target update interval	200	steps
Replay buffer size	5000	episodes
Hidden dimension	128	

C Additional Experimental Analysis

C.1 Task Performance (Q1)

Full learning curves on SMAC-Comm are in Figure 2 of the main text. Per-scenario:

bane_vs_hM: SIGMA converges fastest and reaches the highest win rate. Three Banelings must detonate simultaneously, requiring precise timing. SIGMA's probabilistic communication shares uncertainty about the Hydralisk's position, enabling more reliable synchronization than deterministic baselines.

2o_20b_vs_2r: The performance gap is largest. Only Overseers can see Roaches, so Banelings rely entirely on communication. SIGMA's Bayesian fusion weights confident Overseer messages heavily and down-weights uncertain messages from distant Banelings.

5z_vs_1ul: SIGMA is stable, while TarMAC and MASIA show high variance. Zealots must coordinate kiting and focus fire. SIGMA's uncertainty weighting sends rich messages when the Ultralisk's target is uncertain and compact messages when it is predictable.

Paired t-tests (one-tailed, 5 seeds) confirm SIGMA outperforms LMAC across all scenarios ($p < 0.05$ for SMACv2, LBF, and GRF; $p < 0.01$ for budget and transfer). The smallest margin is on GRF run_pass (about 1 point); SMACv2 gains 3.6–4.2 points, and LBF, GRF 3_vs_1, budget, and transfer show larger gaps. Effect sizes (Cohen's d) range from 0.45 to 2.36, indicating medium-to-large effects. These p-values are reported as descriptive statistics without multiple-comparison correction. The main conclusions rely on consistent gains across benchmarks rather than on any single scenario. Full per-seed statistics, standard deviations, and learning curves for baselines on target maps are included in the released code to further illustrate their adaptation difficulty.

C.2 Communication Efficiency (Q2)

For budget experiments, all methods are retrained from scratch under each budget with default hyperparameters. For SIGMA, latent dimension d is reduced proportionally, so both μ and σ shrink; for baselines, message dimension varies over $\{16, 32, 64, 128\}$. No per-budget tuning is performed.

Message design. We compare probabilistic messages with deterministic μ-only messages and raw observation sharing. As shown in Table 12, probabilistic messages outperform both, confirming that σ provides benefits beyond transmitting more information. Raw observation sharing uses the most

bits but performs worst because it floods the receiver with irrelevant information. Deterministic μ uses fewer bits but cannot perform uncertainty-weighted fusion.

Table 12: Comparison of message designs on 2o_20b_vs_2r. The probabilistic setting corresponds to the default configuration $d = 64$, matching the budget-128 setting in Table 2.

Message Design	**Win Rate (%)**	**Bits/Step**
Raw observation sharing	68.5	16,384
Deterministic μ only	72.7	2,048
Probabilistic (μ, σ)	**84.5**	4,096

Robustness to noise and dropouts. On 2o_20b_vs_2r, with 10% random message drops or Gaussian noise $\sigma = 0.1$ added to transmitted (μ, σ), SIGMA's win rate drops by less than 2% (84.5% to 82.8% and 83.2%). LMAC drops by 5–7%, and TarMAC by 8–10%. Inverse-variance fusion thus provides robustness to unreliable communication.

Bits-per-timestep comparison. Continuous methods count each float as 32 bits. For LMAC, each of 14 discrete symbols counts as $\log_2 K$ bits, a conservative estimate. Table 13 reports win rates under matched bit budgets. The ranking is unchanged: SIGMA is best at every budget, with the largest advantage at low budgets.

Table 13: Win rate (%) under matched bits-per-timestep budgets on 2o_20b_vs_2r.

Bits/Step	**512**	**1024**	**2048**	**4096**
TarMAC	42.1	55.3	65.3	72.8
NDQ	43.5	56.2	66.1	73.5
MASIA	45.8	58.7	68.4	75.2
LMAC	48.2	61.5	71.8	79.4
CommFormer	44.5	57.8	67.1	74.2
SIGMA	**68.4**	**76.2**	**81.9**	**84.5**

Computational overhead. Table 14 summarizes training time. SIGMA adds only modest overhead over QMIX because the VAE encoder is a lightweight MLP and Bayesian fusion adds a small constant cost. The one-time LLM annotation for 5K samples consumes about 72K tokens, roughly $0.25 API cost, and is excluded from per-step cost.

Table 14: Training time on 2o_20b_vs_2r.

Metric	**QMIX**	**LMAC**	**SIGMA**
Training Time (h)	5.4	8.8	6.2

Inference overhead and bandwidth. At execution, SIGMA's per-step cost consists of a forward pass through the VAE encoder (a 3-layer MLP with 128 hidden units) and Bayesian fusion ($O(|\mathcal{C}(i)| \cdot d)$ per agent). The encoder adds negligible latency compared to the Q-network, and fusion requires only element-wise operations over the latent dimension. In terms of bandwidth, under the same budget K floats per agent per step, SIGMA transmits exactly $K/2$ values for μ and $K/2$ for σ, identical to baselines that transmit K floats. The difference is that SIGMA's messages carry uncertainty information, enabling confidence-weighted fusion; it does not use more bandwidth. At the default $d = 64$, this is 128 floats per step, matching the budget-128 setting in Table 2.

C.3 Uncertainty-Aware Communication (Q3)

Validation protocol and correlation. We train a 3-layer MLP with 128 hidden units that takes $(\mu_i, \sigma_i, \tau_i)$ and predicts the global state $\hat{s}$, using MSE loss, Adam at 10^{-3}, and 1000 epochs. Training data are trajectories from the trained SIGMA model paired with simulator ground-truth global states; this decoder is only for evaluation. On 2o_20b_vs_2r, we run 100 evaluation episodes (max 100 steps,

22 agents), yielding about 200,000 agent-timestep pairs. For each sample we record the average $\bar{\sigma}_i$ over latent dimensions and reconstruction error $e_i = \|s - \hat{s}_i\|^2$. The Pearson correlation is $r = 0.89$, with $p < 0.001$ (two-sided t-test, $M - 2$ df). For a Gaussian posterior, differential entropy $H_j = \frac{1}{2}\log(2\pi e\sigma_j^2)$ is strictly increasing in σ_j, while KL divergence is not monotonic (minimized at $\sigma_j = 1$); larger σ thus corresponds to a more informative dimension. Figure 3(b) confirms this monotonic relationship. μ contributes a mean shift while σ controls variance expansion. In our framework, σ is a tractable proxy for posterior spread and is used as a confidence weight during Bayesian fusion. σ reflects posterior spread, while μ carries content; the effective information flow is determined by the fusion weights, not by σ alone. $I(z; o)$ estimated via InfoNCE increases during training but is not monotonic in σ, consistent with the standard IB view.

Controls and robustness. Repeating the procedure for three seeds gives $r \in \{0.87, 0.89, 0.91\}$, and the correlation remains $r > 0.80$ when σ is averaged over only the top-16 latent dimensions. A μ-only decoder with the same architecture and data has higher reconstruction error than the full decoder at all uncertainty levels, and a weaker correlation with σ ($r = 0.61$ vs. 0.89; Table 15). This confirms that σ carries additional uncertainty-relevant information beyond μ. We do not randomize σ at test time, since that would shift the input distribution and confound the comparison; the μ-only decoder is a clean ablation. Controlled occlusion significantly increases σ, while additive observation noise has a smaller effect, indicating that σ primarily encodes uncertainty from partial observability.

Table 15: Control experiment for uncertainty validation.

Decoder Input	**Avg. Reconstruction Error**	**Correlation with σ**
Full (μ, σ)	0.23	0.89
μ only	0.41	0.61

Calibration. σ^2 is produced by the VAE and is not explicitly calibrated. Temperature scaling learns a scalar T on held-out data such that $\tilde{\sigma}^2 = \sigma^2/T$ minimizes the NLL of state reconstruction error under a Gaussian model, without retraining the VAE or policy. Because σ_{fuse} is not fed into the utility network, this scaling affects only the reported uncertainty and calibration metrics, not the policy. As shown in Table 16, uncalibrated SIGMA already improves over constant variance; after scaling, ECE drops from 0.11 to 0.04 and NLL from 1.12 to 0.87. On transfer targets, ECE/NLL improve from 0.13/1.18 to 0.06/0.92 (Table 17), confirming reliability under distribution shift. Calibration failures occur mainly under extreme occlusion where σ is overestimated. Covariance intersection gives comparable ECE/NLL but requires additional tuning; temperature scaling achieves ECE 0.04/NLL 0.87 and 84.5% win rate, versus 0.05/0.91 and 84.1% for covariance intersection, so we adopt the simpler scaling.

Table 16: Calibration metrics on held-out data. Lower is better.

Method	**ECE**	**NLL**
Constant variance	0.13	1.34
SIGMA (uncalibrated)	0.11	1.12
SIGMA (temperature scaled)	**0.04**	**0.87**

Table 17: Calibration on transfer targets.

Setting	**ECE**	**NLL**
Uncalibrated	0.13	1.18
Temperature scaled	**0.06**	**0.92**

A limitation of Bayesian fusion in Eqs. 4–5 is the independence assumption. When agents observe correlated views, this assumption underestimates the fused variance and can lead to overconfidence. Theoretically, for two messages with correlation ρ, the true fused precision is $\sigma_{\text{fuse}}^{-2} = \sigma_i^{-2} + \sigma_j^{-2} - 2\rho\,\sigma_i^{-1}\sigma_j^{-1}$, which is smaller than the independent case when $\rho > 0$, implying a

larger fused variance. Temperature scaling calibrates the reported σ^2 via a single scalar T, reducing measured overconfidence without altering the fused mean or the policy. The fusion weights already down-weight uncertain messages through inverse-variance weighting, so calibration is only needed for reporting. Empirically, temperature scaling yields ECE 0.04 and NLL 0.87, comparable to covariance intersection (ECE 0.05, NLL 0.91) but with lower tuning cost. A learned cross-correlation penalty or covariance intersection with per-pair estimation is left for future work.

The temperature scalar T is learned on a small held-out set with global states available during centralized training, and then fixed during execution, so no global state is required at deployment. Since T only calibrates the reported uncertainty and does not affect action selection, it requires very few held-out samples; in practice, a small calibration set (e.g., a few hundred transitions) suffices, and the resulting ECE is stable across calibration set sizes. This makes the approach practical even when global state is only available during centralized training.

Execution and dynamics. Although the VAE is trained with the reparameterization trick, at execution agents transmit (μ_i, σ_i) and the receiver uses the fused mean μ_{fuse} as a deterministic representation for the Q-network; sampling z at execution reduces win rate by 2–3%. σ is positively correlated with policy entropy ($r = 0.72$). Figure 3(b) tracks σ and differential entropy $H(q_\phi(z|o)) = \frac{1}{2}\log(2\pi e\sigma^2)$ over a trajectory in 2o_20b_vs_2r. Entropy is small when the posterior is concentrated (confident, compressed) and large when it is spread out (uncertain, informative). Similar patterns appear in bane_vs_hM and 5z_vs_1ul, confirming that uncertainty-weighted communication generalizes.

C.4 Ablation Study (Q4)

We ablate LLM variants, fusion methods, value-decomposition backbones, and hyperparameters.

LLM variants. Table 18 compares GPT-5.6 Sol, GPT-5-mini, and Claude annotations. All support strong downstream performance; GPT-5.6 Sol is best, and the gap to smaller LLMs is small. Performance is also stable with 1K annotations, indicating low sensitivity to annotation quality. Importantly, the "w/o LLM" variant in Table 3 retains the same VAE architecture and observation reconstruction loss but removes the semantic branch entirely. Its performance drop therefore isolates the contribution of LLM-provided semantic structure rather than that of an auxiliary reconstruction task, since observation reconstruction alone is already present. The variation across LLMs in Table 18 further indicates that semantic content, not merely an auxiliary loss, drives the improvement. While we do not run a separate random-label control, the consistent gap between LLM-annotated and w/o LLM variants across all three maps suggests that semantic content, rather than a generic auxiliary loss, is the primary driver.

Table 18: Performance with different LLMs for annotation on three SMAC-Comm maps.

LLM	2o_20b_vs_2r	bane_vs_hM	5z_vs_1ul
GPT-5.6 Sol	84.5 ± 2.7	76.6 ± 2.3	93.5 ± 1.6
GPT-5-mini	79.1 ± 3.0	74.2 ± 2.7	90.2 ± 2.2
Claude Sonnet 5	80.4 ± 2.8	75.3 ± 2.5	91.7 ± 2.0

Fusion methods. Bayesian fusion outperforms simple averaging and attention-weighted averaging (Table 19). Simple averaging lets noisy/uncertain messages corrupt the fused representation; attention-based fusion learns weights but does not explicitly model uncertainty, and underperforms especially when observation quality is heterogeneous.

Table 19: Comparison of fusion methods on three SMAC-Comm maps.

Fusion Method	2o_20b_vs_2r	bane_vs_hM	5z_vs_1ul
Bayesian (Full)	84.5 ± 2.7	76.6 ± 2.3	93.5 ± 1.6
Simple averaging	76.3 ± 3.3	71.6 ± 2.9	88.6 ± 2.5
Attention-weighted	78.9 ± 3.1	73.8 ± 2.6	90.7 ± 2.1

Value-decomposition backbones. SIGMA brings consistent gains over VDN, QMIX, and QPLEX (Table 20), confirming that the communication module is agnostic to the CTDE learner. The gain is largest for VDN, whose individual utility representation is weakest.

Table 20: Performance with different value-decomposition backbones on 2o_20b_vs_2r.

Backbone	QMIX	VDN	QPLEX
No Comm	52.3	48.7	55.1
SIGMA	**84.5**	**78.4**	**80.6**

Hyperparameter sensitivity. Latent dimension $d \in \{16, 32, 64, 128\}$: performance improves up to 64 and plateaus, so $d = 64$ balances expressiveness and overhead. KL weight $\beta \in \{0.001, 0.01, 0.1, 1.0\}$: small β under-compresses, large β over-compresses; $\beta = 0.01$ is best, consistent with information bottleneck theory. Slow-update interval $K \in \{10, 30, 50, 100\}$: smaller K lets the VAE over-adapt to RL signals and corrupt semantics, larger K makes it too rigid; $K = 50$ balances stability and adaptability. Semantic loss weight $\alpha \in \{0.1, 0.3, 0.5, 1.0\}$: $\alpha = 0.5$ prevents semantic drift while allowing task adaptation; too large over-constrains the latent space, too small degrades semantics.

C.5 Cross-Task Transfer (Q5)

We train SIGMA on 2o_20b_vs_2r, freeze the VAE and communication protocol, and train a lightweight observation adapter plus new utility/mixing networks on bane_vs_hM and 5z_vs_1ul using 100 target-map episodes. Target maps have observation dimensions 31 and 36 versus 171 in the source; the adapter is a single linear layer $d_{\text{target}} \times d_{\text{source}}$ projecting target observations to the source dimension before the frozen encoder. It is trained with the same reconstruction loss as VAE pretraining, Adam at 10^{-3}, 200 epochs, batch size 128. The adapter only aligns input dimensions and does not relearn latent semantics. LLM annotations capture shared semantics across maps (agent status, relative positions, partial-observability uncertainty), making the VAE latent space task-agnostic. Baselines such as LMAC and TarMAC rely on task-specific message formats and attention statistics, which break when observation dimensions change. The linear adapter only projects target observations to the source dimension and does not learn a new semantic mapping. The transfer setting here focuses on SMAC maps that share entity-level semantics.

Transfer with 100 adapter episodes reaches 74.8% on bane_vs_hM and 81.0% on 5z_vs_1ul, close to from-scratch performance (76.6% and 83.2%) with far fewer interactions (Table 21). From-scratch numbers use the same number of environment steps as the transfer setting.

Table 21: Transfer vs. from-scratch on target maps. Transfer uses 100 adapter episodes; from-scratch trains the full model for the same number of environment steps.

Target Map	Transfer (100 ep.)	From Scratch
bane_vs_hM	74.8	76.6
5z_vs_1ul	81.0	83.2

Adapter capacity and VAE fine-tuning. A linear adapter performs comparably to a 2-layer MLP with 64 hidden units, while zero-padding degrades performance by 2–3% (Table 22), confirming that lightweight linear dimension alignment is sufficient. Fine-tuning the VAE degrades performance by 5–8% compared to freezing (Table 23), because task-specific gradients corrupt pretrained semantics. The slow-update strategy or full freezing preserves task-agnostic semantics and enables effective transfer. The adapter assumes aligned observation semantics; for raw pixels or fundamentally different entity semantics, a learned semantic extractor would be needed.

All transfer results are averaged over 5 seeds; the gaps to baselines are large and consistent across seeds. Full per-seed statistics and standard deviations are included in the released code.

Table 22: Transfer performance (win rate %) with different adapter capacities.

Adapter	bane_vs_hM	5z_vs_1ul
Linear (default)	**74.8**	**81.0**
2-layer MLP	74.5	80.8
Zero-padding	72.1	78.3

Table 23: Transfer performance (win rate %) with frozen vs. fine-tuned VAE.

VAE Setting	bane_vs_hM	5z_vs_1ul
Frozen (default)	**74.8**	**81.0**
Fine-tuned (no slow update)	68.2	75.4

C.6 SCALABILITY (Q6)

Race-specific degradation. Terran needs healing coordination due to Medivacs; SIGMA prioritizes critical uncertainty messages and shows the smallest 5v5-to-20v20 degradation. Protoss requires protecting the high-damage, low-mobility Colossus and coordinating focus fire; the larger 92-dimensional observation space makes communication harder, causing slightly larger degradation. Zerg requires precise Baneling suicide timing; baselines degrade most, while SIGMA remains stable by down-weighting uncertain timing messages via Bayesian fusion.

Training efficiency. Steps to reach 50% win rate grow with team size for TarMAC and LMAC, which must learn additional pairwise weights or protocol rules. For SIGMA, required steps remain roughly constant across 5v5, 10v10, and 20v20, because the VAE and communication protocol are pretrained and Bayesian fusion is permutation-invariant, adding no new parameters as agents increase. The policy still adapts to a larger action space, but the communication module transfers without additional training. All scalability results are obtained under identical training budgets and evaluation protocols; no per-benchmark hyperparameter tuning is performed for any method.

Computational complexity. Bayesian fusion (Eqs. 4–5) requires $O(|\mathcal{C}(i)| \cdot d)$ operations per agent, where d is the latent dimension and $|\mathcal{C}(i)|$ is the number of communicating neighbors. In a broadcast setting, $|\mathcal{C}(i)| = N - 1$, so per-agent fusion scales linearly with team size, but this is inherent to any communication method that aggregates messages from all teammates. The VAE encoder is a lightweight MLP with 128 hidden units, adding negligible inference cost compared to the Q-network. The communication module itself introduces no new parameters as team size grows, which is why steps to 50% win rate remain roughly constant. Total bandwidth scales with the number of communication links; reducing this via sparsity or neighbor selection is an orthogonal direction left for future work.

Prompt scalability. The prompt template consists of three reusable components: task description, state information, and observation information. These are readily available from standard environment documentation or interfaces, and the template structure remains identical across environments; only the placeholders are filled with environment-specific text. No per-environment prompt engineering is required beyond transcribing the observation semantics, which is a one-time offline step. As shown in Table 18, performance is stable across different LLMs and annotation sizes, indicating low sensitivity to annotation quality and prompt wording.

D THE USE OF LARGE LANGUAGE MODELS

Large language models (LLMs) were used in this work for two purposes. First, LLMs were employed for writing assistance, including grammar checking, language polishing, and improving the readability of the manuscript. Second, LLMs were used for generating the semantic annotations in experiments: OpenAI's GPT-5.6 Sol was used exclusively for the one-time offline annotation described in Appendix A.4, and GPT-5-mini and Claude Sonnet 5 were used only in the LLM-variant ablation (Appendix C.4). The LLM was not used for research ideation or for designing the proposed method. The authors take full responsibility for the content of this paper.