technologies

MDPI

Article

Bridging Heterogeneous Agents: A Neuro-Symbolic Knowledge Transfer Approach

Artem Isakov *, Artem Zaglubotskii, Ivan Tomilov, Natalia Gusarova, Aleksandra Vatian * and Alexander Boukhanovsky

Department of Infocommunication Technologies, Information Technologies, Mechanics and Optics University, 197101 Saint-Petersburg, Russia; avzaglubotskii@itmo.ru (A.Z.); ivtomilov@itmo.ru (I.T.); nfgusarova@itmo.ru (N.G.) avbukhanovskii@itmo.ru (A.B.)

* Correspondence: aoisakov@itmo.ru (A.I.); asvatian@itmo.ru (A.V.)

Abstract

This paper presents a neuro-symbolic approach for constructing distributed knowledge graphs to facilitate cooperation through communication among spatially proximate agents. We develop a graph autoencoder (GAE) that learns rich representations from heterogeneous modalities. The method employs density-adaptive *k*-nearest neighbor (*k*-NN) construction with Gabriel pruning to build the proximity graphs that balance local density awareness with geometric consistency. When the agents enter the bridging zone, their individual knowledge graphs are aggregated into hypergraphs using a construction algorithm, for which we derive the theoretical bounds on the minimum number of hyperedges required for connectivity under arity and locality constraints. We evaluate the approach in PettingZoo's communication-oriented environment, observing improvements of approximately 10% in episode rewards and up to 40% in individual agent rewards compared to Deep Q-Network (DQN) baselines, while maintaining comparable policy loss values. The explicit graph structures may offer interpretability benefits for applications requiring auditability. This work explores how structured knowledge representations can support cooperation in distributed multi-agent systems with heterogeneous observations.

Keywords: multi-agent systems; internet of agents; reinforcement learning; representation learning; knowledge graphs; neuro-symbolic artificial intelligence

check for updates

Academic Editors: Martin Valtierra-Rodriguez, Dinh-Thuan Do, Vitor Fialho, Luis Pires, Francisco Rego, Ricardo Santos and Vasco Velez

Received: 11 October 2025
Revised: 14 November 2025
Accepted: 2 December 2025
Published: 4 December 2025

Citation: Isakov, A.; Zaglubotskii, A.; Tomilov, I.; Gusarova, N.; Vatian, A.; Boukhanovsky, A. Bridging Heterogeneous Agents: A Neuro-Symbolic Knowledge Transfer Approach. *Technologies* **2025**, *13*, 568. https://doi.org/10.3390/technologies13120568

Copyright: © 2025 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

The Internet of Agents (IoA) has successfully evolved from the fusion of Internet of Things (IoT) and AI techniques [1,2]. It is plausible to argue [3,4] that one of the key paradigms for modeling and implementing many IoT scenarios is the concept of multi-agent systems (MASs), which treat every device as an agent with its own distinct set of needs, capabilities, and inter-agent relationships. An MAS provides developers with approaches that enable the design of scalable and fault-tolerant IoA systems.

On the one hand, a variety of game theoretic methods decompose multi-agent interactions into sequences of two-player zero-sum games. These methods have evolved to incorporate accurate opponent intention recognition models [5] and sophisticated state-pruning techniques, enabling faster convergence to equilibria [6], which appears to be beneficial in defense strategies and adversarial robustness analysis. On the other hand, Multi-Agent Reinforcement Learning (MARL) [7,8] enables multiple agents to simultaneously engage in general-sum games within a shared environment. Each agent receives

observations, executes actions, and obtains rewards—positive signals that guide learning. Optimizing cumulative rewards across agents directly improves system-level performance metrics such as latency reduction, task completion rates, and resource efficiency, which serves to advance practical multi-agent systems in domains ranging from autonomous logistics to distributed sensing networks.

A central challenge in MARL is learning to cooperate within non-stationary environments and with limited local information [9–11]. The centralized training with decentralized execution (CTDE) paradigm partially mitigates non-stationarity by exploiting global signals during training while relying on local observations during execution [12–14]. However, most CTDE approaches assume fixed communication routes, manually designed message structures or implicit weight exchange, which restricts scalability and interpretability.

Over the past few years, researchers have proposed differentiable communication architectures to improve coordination, including CommNet [15], DIAL [16], TarMAC [17], and more recent transformer- and graph-based models such as CommFormer [18] and MAGNet [19]. While these architectures learn message-passing policies end-to-end, they still rely on implicit topologies that may not capture the evolving semantics of IoA environments.

An alternative direction, actively explored in recent works [20–23], seeks to organize a shared semantic space, or a field of knowledge, typically represented as a knowledge graph in the space of the agents' embeddings. Most authors employ a single rigid knowledge graph that provides communication, which facilitates centralized optimization but makes the system fragile to connectivity disruptions, such as agent dropout or communication failures.

From the perspective of IoA, the design must shift its focus from centralized optimization to sustaining decentralized locally optimal interactions among spatially or semantically proximate agents. Proximity can be defined by physical or logical constraints, for example, by Bluetooth range or smart-home boundaries. To achieve the shift, we propose a two-level hierarchical reinforcement learning (HRL) framework for IoA entities. The first level employs an actor–critic architecture [12,13], in which a centralized critic forms a coordination graph [24,25] that links agents within a controlled detection radius referred to as the bridging zone. The second level uses neuro-symbolic AI principles [26,27] to construct a distributed knowledge graph [28,29] for the agents within the bridging zone. When the agents enter the bridging zone, their individual knowledge graphs representing their local understanding of the environment are connected by hyperedges, forming multiple dynamic knowledge hypergraphs. This structure provides more ample mutual information for coordination and decision-making.

While graph factorization at global, neighborhood, or agent levels has already been studied [30,31], applying such factorization to distributed knowledge graphs seems to remain unexplored.

The first-level problem is common for MARL and can be addressed using existing techniques, including QMIX [31] or attention-based actor–critic methods [12]. Our contribution focuses on addressing the second-level problem as the research gap remains in the construction and aggregation of distributed knowledge graphs that guide interpretable and controlled communication among agents with independent observation buffers operating in partially observable settings. Existing approaches either assume fixed topologies or rely on implicit relationships that lack adaptability to dynamic agent populations with heterogeneous observations.

We introduce a framework that combines (i) a representation learning stage using a GAE and (ii) a hypergraph construction stage that fuses local knowledge graphs under structural constraints.

We first used a large language model (LLM) as a world model to standardize and contextually enrich raw observations from PettingZoo's Multi-Particle Environments (MPEs) suite [32]. We then mapped these vectors to agent knowledge graphs using the proposed GAE model, which extends conventional k-NN graphs with a density scaling factor α and Gabriel pruning threshold β to mitigate over-smoothing and over-squashing [33–35] effects. Next, we theoretically derived the minimum number of bridging hyperedges required and introduced and algorithm to construct a dynamic hypergraph by aggregating distributed agent knowledge graphs. The resulting hypergraph is shared within each bridging zone as a collective observation, informing the policy updates.

Our main contributions are summarized as follows:

1. We introduce a density-adaptive k-NN algorithm with Gabriel pruning that constructs proximity graphs responsive to local point density while maintaining geometric consistency, achieving superior reconstruction fidelity compared to alternatives.
2. We present a hypergraph construction algorithm for aggregating distributed knowledge graphs within bridging zones and derive the theoretical bounds on the number of hyperedges required for connectivity under arity and locality constraints.
3. We empirically demonstrate that our neuro-symbolic approach achieves improvements of approximately 10% in episode rewards and up to 40% in individual agent rewards over DQN baselines, while maintaining comparable policy loss values.

The rest of this paper is organized as follows: Section 2 reviews related work on MARL communication, graph-based architectures, and representation learning. Section 3 introduces theoretical preliminaries including autoencoders, proximity graphs, and reinforcement learning foundations. Section 4 presents our methodology: the multi-agent environment, format-invariant representations via language models, the graph autoencoder architecture, density-adaptive graph construction, bridging hypergraph aggregation with theoretical bounds, and experimental design. Section 5 reports empirical results, and Section 6 summarizes contributions, discusses limitations, and potential future directions.

2. Related Works

Learning to communicate in MARL. Neural approaches that learn what to say include continuous fully shared channels (CommNet) [15] and differentiable discrete channels (DIAL) [16]. Targeted addressing (TarMAC) augments content learning with learned recipients [17]. These methods complement CTDE value factorization techniques (e.g., QMIX) that stabilize learning within non-stationarity [31]. More recent works [18,19] frame communication as a learnable graph, jointly optimizing topology and content within transformer-style architectures or graph attention messaging schemes. Adaptive hierarchical communication structures have also been proposed, which learn hierarchical routing strategies among groups of agents [36]. Acknowledging these advances in end-to-end differentiable communication modeling, we shift our focus to the representation level to provide a modular structured foundation that can augment diverse policy architectures without altering their intrinsic communication dynamics.

Graph-based communication and attention. Representing agents and their interactions as a graph enables selective, relational aggregation. Graph attention networks (GAT) introduce masked, content-dependent neighborhood weighting to graphs [37], with GATv2 improving expressivity via dynamic attention score functions [38]. Surveys regarding the intersection of GNNs and MARL report consistent benefits when message passing respects the relational structure and task context.

Proximity graphs and geometric priors. Classical geometric graphs provide controllable sparsity–locality trade-offs: Delaunay triangulations and Gabriel Graphs are canonical constructions used to retain informative local connections while pruning spu-

rious edges [39,40]. Such priors motivate constructing communication topologies from learned latent coordinates before applying attention-based message passing.

Representation learning. This field aims to obtain representations that are simultaneously comprehensible to neural networks and interpretable for symbolic reasoning [27,41]. The following approaches are the most relevant to our work:

- **Knowledge Graph Embedding (KGE)** [42,43]: Nodes and relations are represented as continuous vectors preserving adjacency and logical patterns. Conventional KGE methods use a full adjacency matrix, which is not scalable or dynamically updateable.
- **Hierarchy-preserving embeddings** [44,45]: These impose a hierarchical structure through ontologies or nested manifolds, predicting connection probabilities without assessing semantic importance. Redundant edges increase computational load and communication cost.

Recent work has also explored knowledge graph construction from heterogeneous data [46], automated pipeline-based approaches for domain-specific KG building [47], and neuro-symbolic methods for creative sequence generation [48].

The GAE approach [49–51] is widely used for representation learning, typically encoding adjacency information and reconstructing node relations. However, existing GAEs do not appear to be directly applicable to our methodology. For example, ref. [49] applied GAE to prerequisite relation discovery, while ref. [42] used it for dimensionality reduction followed by symbolic inference. In most cases, symbolic reasoning is performed separately via domain-specific logic rather than jointly optimized with graph embeddings.

Meanwhile, our proposed GAE integrates structural adaptation and neuro-symbolic embedding, supporting dynamic topology and context-aware reasoning required for multi-agent environments.

Benchmarks. PettingZoo unifies multi-agent APIs and provides MPEs, including benchmark tasks that isolate communication effects [14,32]. These environments provide consistent state–action–reward conventions and reproducibility, facilitating fair comparisons of MARL algorithms.

3. Preliminaries

3.1. Autoencoders

Variational Autoencoder. Variational Autoencoders (VAEs) are a class of generative models that combine the principles of deep learning with probabilistic graphical models [52,53]. The key is to approximate the intractable posterior distribution $p_\theta(z \mid x)$ of latent variables z given data x by introducing an encoder network $q_\phi(z \mid x)$, parameterized with ϕ. The decoder network $p_\theta(x \mid z)$, parameterized with θ, maps latent representations back to the data space.

The training objective maximizes the marginal likelihood of the observed data. Since the marginal log-likelihood $\log p_\theta(x)$ is generally intractable, VAEs optimize the Evidence Lower Bound (ELBO):

$$\log p_\theta(x) \geq \mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x \mid z)] - \mathrm{KL}\big(q_\phi(z \mid x) \,\|\, p(z)\big), \tag{1}$$

where $\mathrm{KL}(\cdot\|\cdot)$ denotes the Kullback–Leibler divergence, and $p(z)$ is typically chosen as a standard Gaussian prior.

In other words, the loss function to be minimized is given by

$$\mathcal{L}(\theta, \phi; x) = -\mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x \mid z)] + \mathrm{KL}\big(q_\phi(z \mid x) \,\|\, p(z)\big). \tag{2}$$

This formulation ensures that the encoder learns a latent distribution close to the prior while the decoder reconstructs the input accurately, balancing regularization and reconstruction.

Conditional Variational Autoencoder. Conditional Variational Autoencoders (CVAEs) extend the VAE framework to model the data conditioned on auxiliary information. Given the input–label pairs (x, y), the encoder is modified to approximate $q_\phi(z \mid x, y)$, while the decoder generates the data conditioned on both the latent variable and the condition, i.e., $p_\theta(x \mid z, y)$. This allows CVAEs to capture complex conditional distributions and makes them suitable for the tasks of image generation with class labels or sequence prediction with contextual variables [54].

The corresponding conditional ELBO is

$$\log p_\theta(x \mid y) \geq \mathbb{E}_{q_\phi(z|x,y)}[\log p_\theta(x \mid z, y)] - \mathrm{KL}\big(q_\phi(z \mid x, y) \,\|\, p(z \mid y)\big), \tag{3}$$

which reduces to the standard VAE objective with an absent conditioning variable y.

3.2. Proximity Graphs

Delaunay Triangulation. Let $V \subset \mathbb{R}^2$ denote a finite set of vertices. A triangulation of V is a subdivision of the convex hull of V into non-overlapping triangles whose vertices belong to V. Among all possible triangulations, the Delaunay Triangulation $\mathcal{D}(V)$ is defined such that no vertex $v \in V$ lies strictly inside the circumcircle of any triangle in $\mathcal{D}(V)$. Similarly, $\mathcal{D}(V)$ maximizes the minimum angle among all triangles, thereby avoiding extremely sharp angles. This optimality makes it fundamental in numerical approximation, mesh generation, and geometric modeling [55].

Delaunay Triangulation is closely related to the Voronoi diagram: two vertices $u, v \in V$ are connected by an edge in $\mathcal{D}(V)$ if and only if their corresponding Voronoi cells share a common boundary. Efficient algorithms exist for constructing $\mathcal{D}(V)$, typically with time complexity $\mathcal{O}(|V| \log |V|)$.

Gabriel Graph. Given a finite set of vertices $V \subset \mathbb{R}^2$, the Gabriel Graph $\mathcal{G}(V)$ is a geometric graph defined on V, in which two distinct vertices $u, v \in V$ are connected by an edge if and only if the closed disk with diameter $\overline{uv}$ contains no other vertex of V. Formally,

$$(u, v) \in E(\mathcal{G}(V)) \iff \nexists w \in V \setminus \{u, v\} \text{ such that } \|w - \tfrac{u+v}{2}\| < \tfrac{1}{2}\|u - v\|, \tag{4}$$

where $\|\cdot\|$ denotes the Euclidean norm.

$\mathcal{G}(V)$ is known to be a subgraph of the Delaunay Triangulation $\mathcal{D}(V)$, since the empty diameter disk condition is stronger than the empty circumcircle condition. The Gabriel Graph is often used in proximity graph theory, spatial network analysis, and wireless communication modeling, as it tends to preserve short-range connections while filtering the redundant edges [56].

***k*-Nearest Neighbor Graph.** For a finite set of vertices $V \subset \mathbb{R}^2$, the k-NN graph $\mathcal{K}_k(V)$ is constructed by connecting each vertex $u \in V$ to its k closest vertices in $V \setminus \{u\}$ according to the Euclidean distance.

The k-NN graph captures local connectivity patterns, provides a flexible and efficient proximity structure, and is widely used in network modeling, where neighborhood size k acts as a tunable parameter controlling sparsity [57].

3.3. Reinforcement Learning

Deep Q-Network. Reinforcement learning problems are often formalized as a Markov Decision Process (MDP), defined by the tuple $(\mathcal{S}, \mathcal{A}, P, r, \gamma)$, where $\mathcal{S}$ and $\mathcal{A}$ denote the

state and action spaces, P the transition probability, r the reward function, and $\gamma \in [0, 1)$ the discount factor. The action–value function $Q^{\pi}(s, a)$ under policy π is defined as

$$Q^{\pi}(s, a) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^{t} r(s_t, a_t) \,\middle|\, s_0 = s, a_0 = a, \pi\right]. \tag{5}$$

The optimal action–value function satisfies the Bellman optimality equation:

$$Q^{*}(s, a) = \mathbb{E}_{s' \sim P}\left[r(s, a) + \gamma \max_{a'} Q^{*}(s', a')\right]. \tag{6}$$

DQN approximates $Q(s, a; \theta)$ with a deep neural network parameterized by θ. To stabilize learning, DQN employs experience replay and a separate target network with parameter θ^{-}. The loss minimized at iteration i is

$$L_i(\theta_i) = \mathbb{E}_{(s,a,r,s') \sim \mathcal{D}}\left[\left(y_i - Q(s, a; \theta_i)\right)^2\right], \tag{7}$$

with target

$$y_i = r + \gamma \max_{a'} Q(s', a'; \theta^{-}). \tag{8}$$

Double Deep Q-Network. DQN tends to overestimate action values because the same network is used for action selection and evaluation. Double DQN (DDQN) addresses this by decoupling the above roles [58]. Specifically, the next action is selected with the online network θ but evaluated using the target network θ^{-}:

$$y_i = r + \gamma Q\left(s', \arg\max_{a'} Q(s', a'; \theta)\,;\, \theta^{-}\right). \tag{9}$$

This simple modification yields more accurate value estimates and improves stability in many environments.

4. Materials and Methods

4.1. Multi-Agent Environment

To investigate the role of communication in groups, we used the PettingZoo of Farama Foundation (Washington, DC, USA), a commonly adopted nonprofit provider of open source MARL benchmarks.

PettingZoo provides a suite of environments for analyzing algorithms that optimize the collective behavior of the settings. Agents' observations are determined entirely by their joint actions. Fully transparent environments, for instance, Stanford Intelligent Systems Laboratory (SISL) *multiwalker_v9*, grant each agent access to a complete vector of observations describing the system state. Moreover, certain game environments, including Butterfly *cooperative_pong_v5*, deliberately restrict what agents can observe, thereby casting the optimization task into a partially observable environment. Nevertheless, agents continue to depend on observation vectors predefined by the environment's configuration.

MPE, a submodule of PettingZoo, introduces an alternative approach by replacing fixed observation vectors with dynamically generated ones that depend on the responses of other agents. This design emphasizes the importance of the observation vector composition as a key factor in multi-agent coordination. The notable examples of such environments within MPE include *simple_speaker_listener_v4* and *simple_reference_v3*.

The simple_speaker_listener_v4 environment features two asymmetric agents and three color-coded targets, red (t1), green (t2), and blue (t3), as presented in Figure 1.

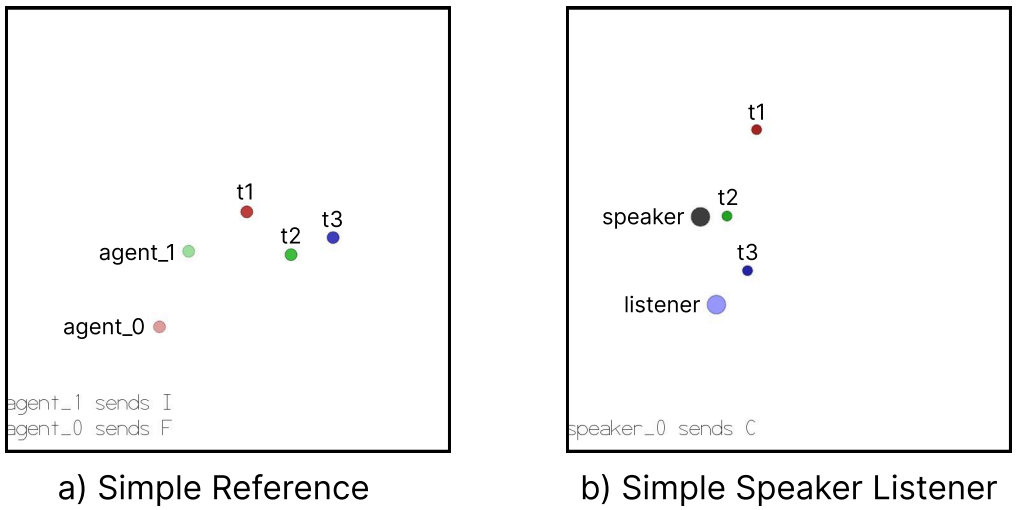


Figure 1. PettingZoo's MPEs: (**a**) Simple Reference and (**b**) Simple Speaker Listener.

The speaker knows the target color for the current game episode and sends a discrete message to guide the listener. The listener perceives its movement direction, the relative distances to targets, and the speaker's messages and can step horizontally, vertically, or not move at all (see Figure 2).

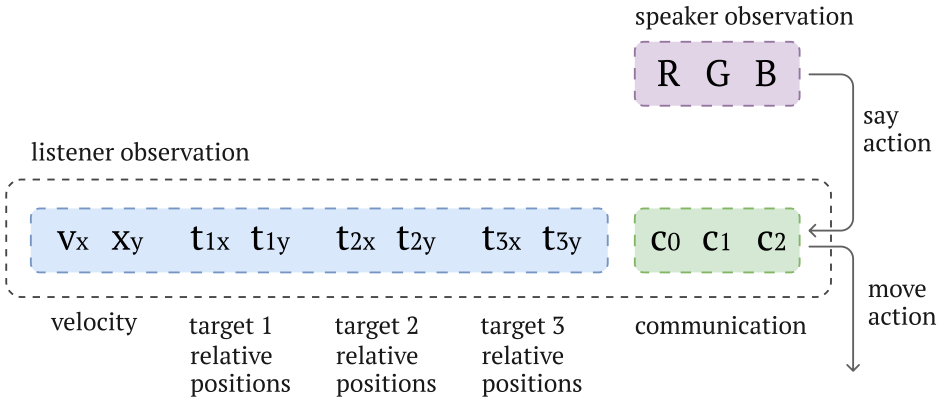


Figure 2. Speaker and listener agents' observation space.

The same applies to the simple_reference_v3 environment, except that each agent simultaneously assumes the roles of both the speaker and the listener. In this environment, each agent knows the direction in which the other agent should move, but lacks the knowledge of its own target.

In *simple_reference_v3*, both agents receive rewards. The closer the agent is to its target, the greater the reward it receives. In *simple_speaker_listener_v4*, the speaker's reward equals the listener's reward. An episode ends either when the listener reaches the target or when a step limit is reached.

As the number of active agents within the environment increases, the overall action space expands with the Cartesian product of the sets of individual agent actions. Combined with the inherent stochasticity of agent policies, this increased complexity compounds reproducibility challenges related to the unstable convergence in reinforcement learning algorithms and heightened sensitivity to implementation details. To examine the role of communication decoupled from reciprocal navigation in a controlled experimental setting conducive to reliable evaluation, our study focused on the *simple_speaker_listener_v4* environment.

4.2. Format Invariance

In agent-to-agent systems, the most prevalent forms of information include continuous signals, interval-based measurements, and categorical levels. When humans are incorporated into interactions with such systems, the range of information modalities expands further to include inputs such as audio commands and visual data (e.g., images, point clouds, and related representations).

Similarly, simulation environments such as *simple_speaker_listener_v4* often comprise heterogeneous agents that differ in the nature of their observations.

To establish a foundation for scaling our game-based example to industry-specific applications, we employed the publicly available GPT-2 under the MIT license to vectorize input representations, irrespective of their original modality.

The listener agent is characterized by an observation space of dimensions $(1, 11)$. Each element is represented as a floating-point number. To process these observations, each float was first converted into a string, as illustrated in Figure 3, and subsequently tokenized.

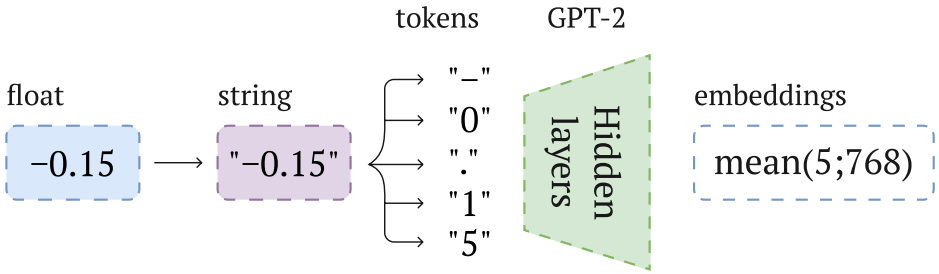


Figure 3. Standardized vectorization procedure for input data.

The resulting tokens, including numeric digits, signs, and decimal points, were passed to GPT-2 to extract vector representations of shape $(11, n, 768)$, where n denotes the number of tokens generated from the string. These vectors were then averaged over n to obtain the shape of $(11, 768)$.

The target label could also be represented in the form of an image. In this case, the preprocessing would remain largely unchanged, although handling this representation would require a more expressive, next-generation multimodal language model, such as GPT-4 mini.

4.3. Autoencoder

Spatial representation. To partition the agents' observations into components that can be recombined to provide new topology and semantic information, we map them into a graph modality. Thus, each element of size $(1, 768)$ from the observation vector $(n, 768)$, where n varies according to the agent's observation size, is fed into the input of a graph autoencoder and mapped to a triplet (x, y, z) of size $(1, 3)$, yielding $(n, 3)$. The first two components, x and y, specify the node's position, while z serves as the node's feature (see Figure 4).

Creating a set of n nodes and connecting them into a complete graph yields a graph-based knowledge representation of the original observation vector. A set of nodes is also referred to as a cloud of points in surface reconstruction problems; this point cloud is combined into a graph in a way that directly impacts the fidelity of the reconstruction, as will be discussed below (see Section 5).

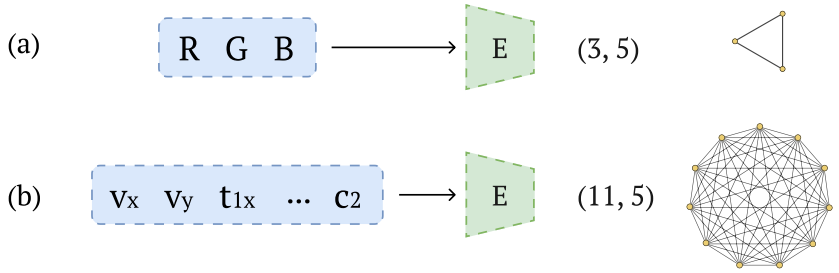


Figure 4. Input transformation with graph autoencoder for the (**a**) speaker observation vector and (**b**) listener observation vector.

Latent Space Regularization. The latent space of a conventional autoencoder has no well-defined structure. In representation learning; however, regularizing the latent space is a crucial step for ensuring meaningful and consistent output.

One approach to regularizing the structure of the latent space is to incorporate label information into the input data, as described for the CVAE in Section 3.1. Gray encoding is a binary numeral system in which consecutive values differ by only one bit, unlike standard binary encoding in which multiple bits can change at once.

This property ensures that semantically related information channels, for example, the projections of a velocity vector onto the x- and y-axes (see Figure 5), are positioned in close proximity within the latent space of the autoencoder network, which is time- and order-invariant, a common prerequisite among real-time applications.

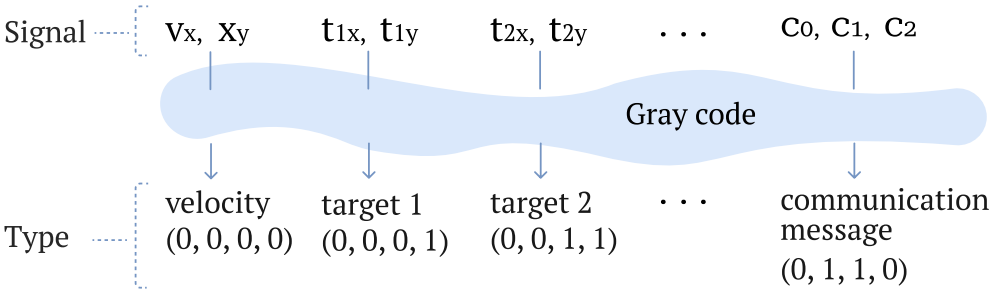


Figure 5. Gray and binary encoding comparison.

Based on the number of unique information channels in our environment, we determined that four bits are sufficient. Consequently, after concatenating the gray encoding with a vector of size $(1, 768)$, the resulting representation has dimensions $(1, 772)$. It is also important to note that the encoding facilitated functional interpretability at the level of a machine learning engineer, particularly during the reverse deployment and analysis of training logs.

Another approach to imposing regularization on the latent representation encoded by a graph is to augment the autoencoder's loss function with an additional penalty term. In our formulation, this regularization term corresponded to the sum of the absolute values of the graph's edge lengths.

Training. The proposed autoencoder employed a three-layer MLP with ReLU activations to map input observations to a k-dimensional latent representation, where the first two dimensions served as spatial coordinates and the remaining $k - 2$ dimensions constituted node features. Graph construction applied parameterizable criteria to the set of nodes, yielding connectivity patterns that encode spatial and semantic relationships. Two parallel GATv2 branches with residual connections processed the resulting knowledge graphs for reconstruction, terminating in specialized heads for categorical and continuous variable

prediction, enabling the model to capture both the discrete and continuous components of an input $(1, 772)$ observation vector (see Figure 6).

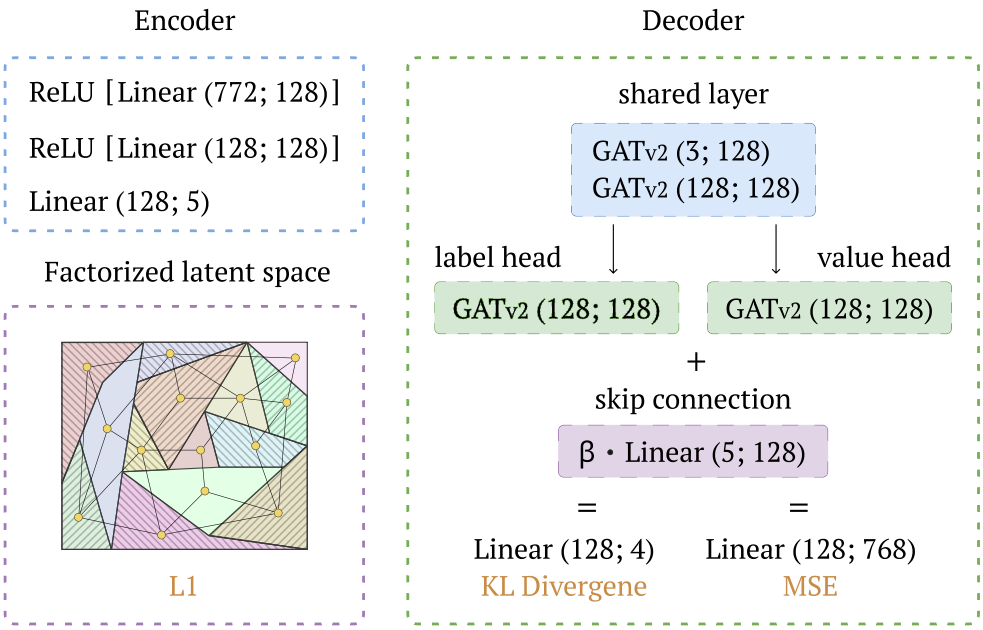


Figure 6. Graph autoencoder architecture.

The loss function used in the training process extended the VAE and CVAE loss function described in Equations (2) and (3) in Section 3.1 so that the reconstruction loss is represented here as the sum of the first two components:

$$\mathcal{L}_{total} = w_1(t)\mathcal{L}_{KL} + w_2(t)\mathcal{L}_{MSE} + \lambda\mathcal{L}_{L_1} \tag{10}$$

where $\mathcal{L}_{KL}$ is the Kullback–Leibler divergence between the predicted distribution and the softmax normalized gray encoded ground truth labels, $\mathcal{L}_{MSE}$ denotes the Mean Squared Error between the predicted and actual feature values, and $\mathcal{L}_{L_1}$ corresponds to the sum of absolute values applied to the edge distances. The time-dependent weighting coefficients $w_1(t)$ and $w_2(t)$ are inversely related, with $w_1(t) + w_2(t) = 1.0$. The hyperparameter λ controls the relative contribution of the edge regularization term to the overall loss.

4.4. Knowledge Graph Construction

The fidelity of reconstructing the original observation vectors depends on the edge selection mechanism. Conventional k-NN approaches employ fixed global connectivity parameters, which fail to adapt to local density variations, while geometric methods, such as Delaunay or Gabriel graphs, impose rigid spatial constraints that may fragment sparse regions, see Section 3.2, Equation (4). To overcome these limitations, we introduce a density-adaptive neighborhood selection strategy augmented with geometric pruning, thereby ensuring global connectivity, local density awareness, and geometric consistency within the constructed graphs.

For a set of vertices $V = \{v_1, \ldots, v_n\} \subset \mathbb{R}^d$ with base neighborhood size k, the local density at point v_i is estimated as

$$d_i = \frac{1}{\frac{1}{k}\sum_{j=1}^{k} \|v_i - u_j\| + \varepsilon} \tag{11}$$

where $u_1, \ldots, u_k$ represent the k nearest neighbors of v_i (excluding v_i itself), and $\varepsilon > 0$ prevents numerical instability from division by zero.

The global median density $\tilde{d} = \text{median}(d_1, \ldots, d_n)$ serves as a normalization reference. The adaptive neighborhood size for point v_i is computed as

$$k_i^{\text{adapt}} = \left\lfloor k \cdot \left(\frac{d_i}{\tilde{d}} \right)^{\alpha} \right\rfloor, \quad k_i^{\text{adapt}} \in [k_{\min}, k_{\max}] \tag{12}$$

The parameter $\alpha > 0$ controls the density sensitivity, while bounds $k_{\min}$ and $k_{\max}$ prevent node isolation and excessive connectivity, respectively.

To balance geometric locality with connectivity preservation, a β-relaxed Gabriel criterion is applied (see Equation (4), Section 3.2). An edge (v_i, v_j) is retained if

$$\|v_k - m_{ij}\| \geq \beta \cdot \frac{\|v_i - v_j\|}{2} \quad \forall v_k \in \mathcal{P} \setminus \{v_i, v_j\} \tag{13}$$

where $m_{ij} = \frac{v_i + v_j}{2}$ denotes the edge midpoint, and $\beta \in (0, 1]$ controls relaxation intensity. The classical Gabriel test corresponds to $\beta = 1$, while $\beta < 1$ permits the retention of additional edges.

The complete procedure is formalized in Algorithm 1.

Algorithm 1 Density-Adaptive k-NN Graph with Gabriel Pruning

Require: Set of vertices $V = \{v_1, v_2, \ldots, v_n\} \subset \mathbb{R}^d$
Require: Nearest neighbors $k \in \mathbb{N}$, default = 6
Require: Density scaling factor $\alpha \in \mathbb{R}^+$, default = 1.5
Require: Gabriel threshold $\beta \in (0, 1]$, default = 1.0
Ensure: Edge set $E \subseteq V \times V$
  Construct KD-tree $\mathcal{T}$ from V
  for $i = 1$ to n **do**
    $\{q_1, \ldots, q_k\} \leftarrow k\text{-NN}(v_i, \mathcal{T}) \setminus \{v_i\}$ ▷ k-Nearest Neighbors
    $d_i \leftarrow \frac{1}{\frac{1}{k}\sum_{j=1}^{k} \|v_i - q_j\| + \epsilon}$ ▷ Local density
  end for
  $\tilde{d} \leftarrow \text{median}(\{d_1, \ldots, d_n\})$ ▷ Median density
  Initialize $E \leftarrow \emptyset$
  for $i = 1$ to n **do**
    $k_i^{\text{adapt}} \leftarrow \left\lfloor k \cdot \left(\frac{d_i}{\tilde{d}} \right)^{\alpha} \right\rfloor$ ▷ Adaptive k
    $k_i^{\text{adapt}} \leftarrow \max(3, \min(k_i^{\text{adapt}}, 2k))$ ▷ Bound k
    $\mathcal{N}_i \leftarrow k_i^{\text{adapt}}\text{-NN}(v_i, \mathcal{T}) \setminus \{v_i\}$
    for each $v_j \in \mathcal{N}_i$ **do**
      if IsGabrielEdge(v_i, v_j, V, β) **then**
        $E \leftarrow E \cup \{(v_i, v_j)\}$
      end if
    end for
  end forreturn E

The initial density computation utilizes KD-tree structures for efficient nearest neighbor queries. Following the median density calculation, adaptive neighborhood sizes are determined and clamped to specified bounds. The β-relaxed Gabriel criterion is then applied to candidate edges from each point's adaptive neighborhood.

This hybrid approach provides tunable control over both the density adaptation through parameter α and the geometric pruning through parameter β, resulting in a unified framework for proximity graph construction that addresses the density–geometry trade-off inherent in existing methods.

4.5. Hypergraph Aggregation

The centers of the knowledge graphs produced with Algorithm 1 are translated within the game environment relative to the positions of their corresponding agents, after which the resulting m components are integrated into a hypergraph.

Let $m \geq 2$ and let $V_1, \ldots, V_m$ be a pairwise-disjoint finite set of vertices $V_i \subset \mathbb{R}^2$. The undirected symmetric k-NN graph on V_i is $G_k(V_i) = (V_i, E_i)$, where $\{u, v\} \in E_i$ if u is among the k-NN of v in V_i or vice versa. A hypergraph $H = (V, \mathcal{E})$ has a 2-section $[H]_2$ with $\{u, v\} \in [H]_2$ if some $e \in \mathcal{E}$ contains both u and v. A bridging hyperedge $e \subseteq V = \bigcup_{i=1}^{m} V_i$ intersects vertices from at least two distinct V_i. See all definitions in Appendix A.1.

We aim to connect the m disjoint k-NN graphs by adding minimal bridging hyperedges while satisfying the set properties (see Appendix A.2). We assume each $G_k(V_i)$ is in its general position and the feasibility graph F on $\{1, \ldots, m\}$ is connected. Here, indices $i, j \in \{1, \ldots, m\}$ label the base components or agent knowledge graphs.

Let $t^\star$ be the minimum number of bridging hyperedges needed. Under Assumptions A1–A3 (see Appendix A.3), we then derive the tight bounds as follows:

Theorem 1 (Bridging bounds)**.**

$$\left\lceil \tfrac{m-1}{r-1} \right\rceil \;\leq\; t^\star \;\leq\; m-1$$

Proof of Theorem 1. Let C be the number of connected components in the 2-section induced by the m blocks before adding any bridging hyperedges; initially, $C = m$.

Lower bound. Suppose a bridging hyperedge e intersects with q distinct components. Then, e can reduce C by at most $q - 1$. Since $|e| \leq r$, necessarily $q \leq r$, so each hyperedge reduces C by at most $r - 1$. To achieve $C = 1$ from $C = m$, we need a total decrease of $m - 1$, and hence, at least $\lceil (m-1)/(r-1) \rceil$ hyperedges. **Upper bound.** Since the feasibility graph F is connected, let T be any spanning tree of F. For each edge $\{i, j\} \in T$, pick feasible representatives $u \in V_i$ and $v \in V_j$ (with $\|u - v\| \leq 2\rho$ if locality is enforced) and add the bridging hyperedge $\{u, v\}$. Each addition reduces C by one, so after $m - 1$ additions, the 2-section is connected. Thus, $t^\star \leq m - 1$. **Tightness.** (i) If geometry allows a single location to meet r components, then $\lceil (m-1)/(r-1) \rceil$ star-like hyperedges can be created, each containing an anchor vertex and up to $r - 1$ vertices from previously unconnected components. This determines the lower bound. (ii) Conversely, let us suppose the m components are arranged in a chain under locality such that any radius-ρ disk intersects the vertices from at most two components while adjacent components are feasible. Then, every bridging hyperedge can reduce C by at most 1, forcing $t^\star \geq m - 1$, and the tree construction is optimal. □

A geometry-aware refinement, accounting for effective arity under locality constraints, is provided in Appendix A.4.

Algorithm 2 returns a hypergraph that has at most $m - 1$ bridging hyperedges and achieves exactly $\lceil (m-1)/(r-1) \rceil$ bridging hyperedges when geometry permits consistent $(r-1)$-way merges.

Algorithm 2 Bridging Hypergraph

Require: Pairwise-disjoint finite set of vertices $V_1, \ldots, V_m \subset \mathbb{R}^2$
Require: Nearest neighbors $k \geq 1$, default = 3
Require: Arity cap $r \geq 2$, default = 4
Require: Radius $\rho > 0$, default = 1.5
Require: Degree cap $L \in \mathbb{N}$, default = 3
Ensure: Base k-NN graphs are connected
Initialize $V \leftarrow \bigcup_{i=1}^{m} V_i$; $\mathcal{E} \leftarrow \varnothing$.
for $i = 1$ to m **do** ▷ retain base k-NN graphs as size-2 hyperedges
build $G_k(V_i) = (V_i, E_i)$; $\mathcal{E} \leftarrow \mathcal{E} \cup \{\{u, v\} : \{u, v\} \in E_i\}$.
end for
Build feasibility matrix $\mathsf{feasible}[i, j] \leftarrow$ **true** if $\exists\, u \in V_i, v \in V_j$ with (**no** radius constraint) or $\|u - v\| \leq 2\rho$.
Initialize a disjoint-set union (DSU) over components $\{1, \ldots, m\}$.
Choose an anchor component $a \in \{1, \ldots, m\}$ with maximum number of feasible neighbors.
$S \leftarrow \{a\}$; maintain per-vertex load counters to enforce degree cap L.
while $|S| < m$ **do**
$\mathsf{CAND} \leftarrow \{j \notin S : \mathsf{feasible}[a, j]\}$ filtered to respect load cap L.
Select $J \subseteq \mathsf{CAND}$ with $|J| \leq r - 1$ maximizing a tie-break score (e.g., nearest to a, balances loads, boosts global coverage).
if $J \neq \varnothing$ **then** ▷ $(r-1)$-way merge when geometry allows
choose representatives $u_a \in V_a$ and $v_j \in V_j$ for $j \in J$ minimizing $\max_{j \in J} \|u_a - v_j\|$;
add hyperedge $e \leftarrow \{u_a\} \cup \{v_j : j \in J\}$ to $\mathcal{E}$; increment loads of involved vertices;
for $j \in J$ **do**
union DSU(a, j);
end for
$S \leftarrow S \cup J$.
else ▷ fallback ensures progress along a spanning tree of F
pick any $j \notin S$ with $\mathsf{feasible}[a, j] =$ **true**;
choose $(u, v) \in V_a \times V_j$ minimizing $\|u - v\|$; add hyperedge $\{u, v\}$ to $\mathcal{E}$; union DSU(a, j);
$S \leftarrow S \cup \{j\}$.
end if
Optionally reset $a \leftarrow \arg\min_{i \in S}$ (estimated future cost) or the node in S with most feasible neighbors in $\overline{S}$.
end while
return $H = (V, \mathcal{E})$.

The algorithm maintains correctness by construction: the base graphs are preserved as size-2 hyperedges, connectivity follows from the union–find merging process, and constraints are enforced during candidate selection. The edge count bound results from each iteration, reducing the component count by at least one, which requires at most $m - 1$ iterations, while optimal merges achieve the theoretical lower bound.

4.6. Experiment Design

We used the environment described in Section 4.1 with the step limit set at 25. When the step limit is reached, the game episode ends, even if the goal has not been achieved. The game may also end earlier than after 25 steps, making the actual number of game episodes variable given a fixed number of global training steps for the RL algorithm.

To standardize all input data and enrich the hidden representations, we leveraged the capabilities of the large-scale GPT-2 language model (MIT license). To extract the embeddings for multinomial input tokens, we applied mean pooling, as described in Section 4.2.

After collecting 10,000 observations from the environment, we constructed a dataset for training the GAE model, the operational principles of which are detailed in Section 4.3.

Using randomly sampled batches of 128 instances, the GAE model was trained to project vectors of continuous values onto graph structures, with the edges produced according to Algorithm 1, as described in Section 4.4. The training objective, defined in Equation (10), consists of a weighted sum of reconstruction and regularization loss functions, with the corresponding weights linearly annealed over 30 epochs. Gradient clipping was applied with a maximum norm of 1.0. The Adam optimizer was employed with a learning rate of 2.5×10^{-3}, and a *ReduceLROnPlateau* scheduler was utilized with a reduction factor of 0.5 and a patience of 5 epochs.

During the training of the reinforcement learning agents, the weights of the GAE model were kept frozen. When the agents entered the bridging zone, their observation vectors were provided as inputs to the GAE. The resulting graphs were subsequently translated according to the agents' positions in the environment. Following the constraints outlined in Theorem 1 from Section 4.5, Algorithm 2 was then employed to construct a shared knowledge hypergraph.

The graph-derived embeddings produced with the GAE were concatenated with the agent's raw observation vectors to form an augmented state representation, which was then passed to the DQN online and target networks. Specifically, the GAT-based hypergraph handler consisted of three GATv2Conv layers with two attention heads in the first two layers and one head in the final layer, processing 3-dimensional node features through a hidden size of 64 to produce a 128-dimensional graph-level representation via global sum pooling. This graph embedding was then combined with a 14-dimensional raw observation vector using a fixed skip connection with the weight $\alpha = 1.0$, while the raw observations were first projected to 128 dimensions through a learned linear transformation. The resulting 128-dimensional augmented state was processed through a multi-layer perceptron with ReLU activations and dropout regularization. The speaker q-network used a three-layer MLP with the layer decreasing from 128 down to the action dimension, while the listener q-network employed a deeper four-layer architecture decreasing from 256 down to the action dimension. Both networks used orthogonal weight initialization and a dropout rate of 0.1 between the hidden layers for improved training stability. The objective function optimized by these networks is defined in Equation (7) above, along with the complementary reinforcement learning formulation and the related training details.

Training protocol. To ensure that any observed performance difference stemmed solely from the inclusion of hypergraph features, both solutions were trained under identical conditions:

- Optimizer: Adam with a learning rate of 5×10^{-4}.
- Training steps: 10,000 environment interactions.
- Replay: Shared buffer capacity of 51,200 transitions; batch size of 128; learning starts after 1000 steps with one update every 5 steps.
- Targets: Soft target updates every 100 steps with rate $\tau = 0.01$.
- Discount and rewards: $\gamma = 0.99$; per-agent rewards are clipped to $[-\sqrt{2}, \sqrt{2}]$ before computing TD targets.
- Exploration: ϵ-greedy linearly annealed from 1.0 to 0.05 over the first 10% of the training steps.
- Regularization: Dropout rate of 0.1; gradient is clipped with a maximum norm of 1.0; orthogonal weight is initialized with a gain of $\sqrt{2}$.
- Loss function: Smooth Huber loss for both agents.

Hyperparameter selection. Hyperparameters for both algorithms were selected using the Tree-Structured Parzen Estimator implemented in Optuna 4.5.0 (MIT license) [59].

Reproducibility. To ensure reproducibility, we conducted experiments across multiple random seeds and reported the mean and one standard deviation for all metrics.

Deterministic behavior was enforced by setting random seeds for PettingZoo 1.24.3 (MIT license), Python 3.12 (PSF license), NumPy 2.2.5 (BSD-3 license), PyTorch 2.7.0 (BSD license), and PyTorch Geometric 2.6.1 (MIT license), with deterministic cuDNN operations possible where applicable.

Computational Environment. Intel Xeon 2.20 GHz, NVIDIA Tesla V100, 16 GB of RAM, Ubuntu 22.04.

5. Results and Discussion

Figure 7 shows examples of the comparison algorithms in constructing a graph based on a given set of nodes.

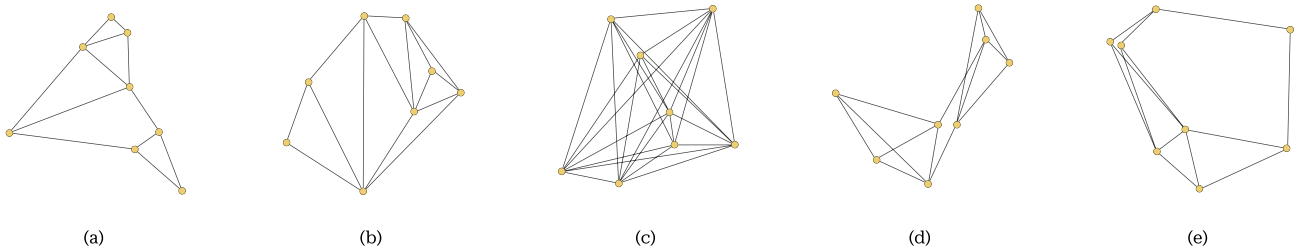


Figure 7. Construction algorithms applied to a point cloud: (**a**) Gabriel graph, (**b**) Delaunay Triangulation, (**c**) complete graph, (**d**) k-NN graph, (**e**) k-NN graph with Gabriel pruning ($k = 6$, $\alpha = 1.5$, $\beta = 1.0$).

As shown in Figure 7e, Algorithm 1 generates a noticeably sparser graph in high-density regions, while introducing additional edges to maintain global connectivity.

Table 1 presents the empirical distribution of component-wise errors.

Table 1. Loss components across graph algorithms at the final epoch (epoch = 30), averaged over 100 random seeds.

Algorithm	$\mathcal{L}_{KL}$	$\mathcal{L}_{MSE}$	$\mathcal{L}_{L_1}$	$\mathcal{L}_{\text{total}}$
k-NN + pruning	$3.6 \times 10^{-3} \pm 2.8 \times 10^{-3}$	$\mathbf{3.8} \times 10^{-2} \pm \mathbf{3.1} \times 10^{-2}$	$\mathbf{7.2} \times 10^{-3} \pm \mathbf{1.0} \times 10^{-3}$	$\mathbf{4.9} \times 10^{-2} \pm \mathbf{3.2} \times 10^{-2}$
Gabriel	$4.0 \times 10^{-3} \pm 3.1 \times 10^{-3}$	$4.9 \times 10^{-2} \pm 4.8 \times 10^{-2}$	$1.1 \times 10^{-2} \pm 1.4 \times 10^{-3}$	$6.4 \times 10^{-2} \pm 5.0 \times 10^{-2}$
Complete	$6.6 \times 10^{-3} \pm 3.7 \times 10^{-3}$	$4.7 \times 10^{-2} \pm 2.9 \times 10^{-2}$	$3.3 \times 10^{-2} \pm 5.8 \times 10^{-3}$	$8.6 \times 10^{-2} \pm 3.2 \times 10^{-2}$
Delaunay	$\mathbf{3.5} \times 10^{-3} \pm \mathbf{2.1} \times 10^{-3}$	$4.2 \times 10^{-2} \pm 3.2 \times 10^{-2}$	$1.7 \times 10^{-2} \pm 2.4 \times 10^{-3}$	$6.2 \times 10^{-2} \pm 3.3 \times 10^{-2}$
k-NN	$3.7 \times 10^{-3} \pm 2.5 \times 10^{-3}$	$5.9 \times 10^{-2} \pm 5.4 \times 10^{-2}$	$1.7 \times 10^{-2} \pm 2.2 \times 10^{-3}$	$8.0 \times 10^{-2} \pm 5.4 \times 10^{-2}$

Note: The table presents the mean $\pm$ one standard deviation. The best results among the compared algorithms are indicated in bold. The mean difference in the total loss between k-NN + pruning and its closest competitor, Delaunay, was 1.38×10^{-2} with a 95% bootstrap confidence interval of $[4.6 \times 10^{-3}, 2.31 \times 10^{-2}]$.

The multi-component loss function $\mathcal{L}$ evaluates the observation label recovery using KL divergence, while continuous values are assessed via the MSE. An additional L1 regularization term penalizes the total weight of edges in the graph.

Figure 8 illustrates various instances for constructing a hypergraph using Algorithm 2, corresponding to various arity constraints, with the number of hyperedges that lie within the bounds established by Theorem 1.

Varying the hyperedge arity constraint captures the levels of relational complexity in a system, from pairwise relationships to higher-order interactions.

Figure 9 presents the resulting knowledge graphs generated with Algorithm 2 for both the speaker and the listener agents within the *simple_speaker_listener_v4* environment.

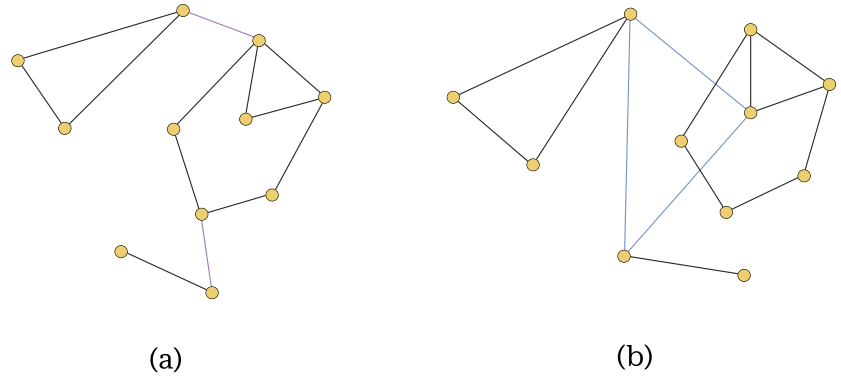


Figure 8. Hypergraphs constructed using Algorithm 2 with various arity constraints: (**a**) $r = 2$; (**b**) $r = 3$.

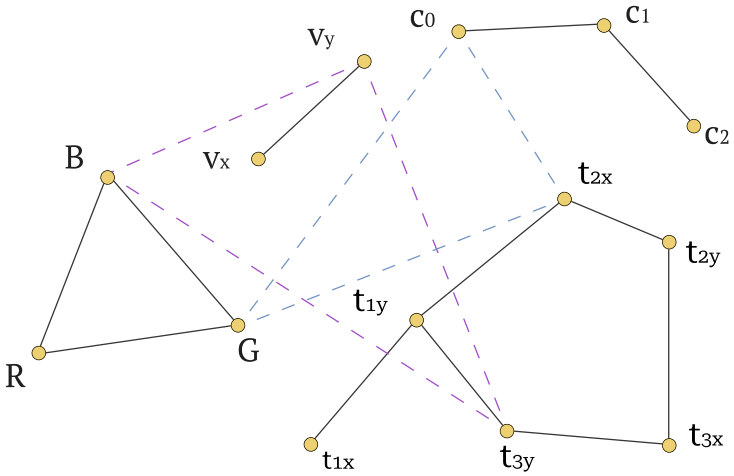


Figure 9. Knowledge hypergraph produced with Algorithm 2.

When the message passes in the GAT layers, the information propagates across the nodes via attention-weighted aggregation, where the learned attention coefficients modulate the strength of communication between the neighboring nodes.

We assessed both the agent-specific metrics (speaker and listener policy losses) and the system-level performance (agent and episode rewards). Figure 10 presents the learning curves recorded.

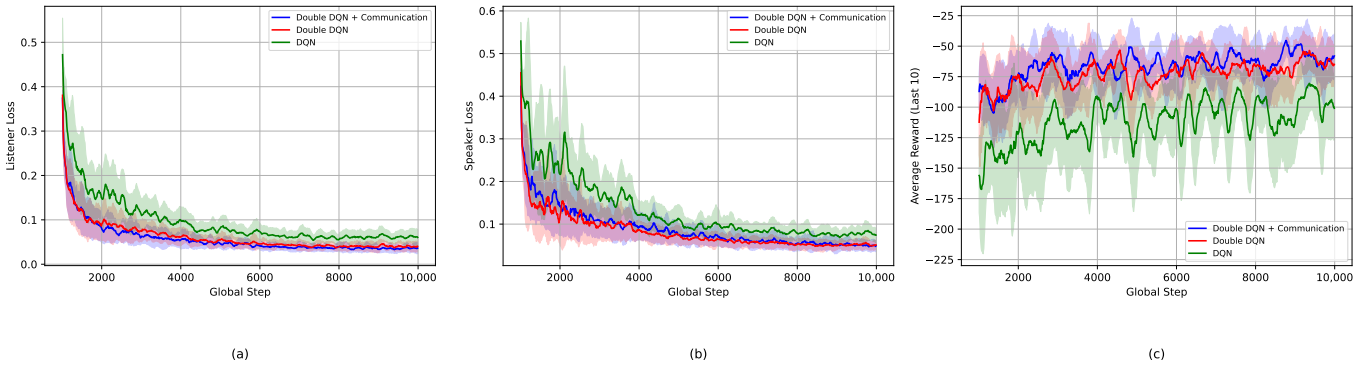


Figure 10. Training dynamics of DQN agents with and without the graph communication mechanism: (**a**) listener loss, (**b**) speaker loss, and (**c**) rolling average over the 10 most recent episodes for episode rewards.

The speaker error decreases at a slower initial rate, but ultimately converges to lower values. Moreover, the listener error exhibits a faster decay and then also attains smaller

values. The rolling average of episode rewards highlights the systematic advantage of incorporating graph communication over the DQN baseline.

Table 2 provides a comprehensive comparison of the return metrics across both configurations.

Table 2. Return comparison.

Metric	Baseline + Comm.	DDQN	DQN
Episode Reward			
Overall avg. reward	$\mathbf{-67.06 \pm 58.18}$	-76.06 ± 63.42	-173.12 ± 97.92
Best step over all seeds	$\mathbf{-41.63 \pm 39.96}$	-53.59 ± 49.69	-106.47 ± 52.17
Agent Reward			
Overall avg. reward	$\mathbf{-5.54 \pm 5.70}$	-6.51 ± 6.33	-16.46 ± 9.55
Best step over all seeds	$\mathbf{-2.48 \pm 2.35}$	-4.18 ± 4.59	-10.93 ± 6.67

Note: The table presents the values of the mean $\pm$ one standard deviation. The best results among the compared algorithms are in bold. The mean difference in the overall average episodic reward between Baseline + Comm. and its closest competitor (DDQN) was 7.24, with a 95% bootstrap confidence interval of [1.03, 13.91]. For the overall average agent reward, the mean difference was 0.81, with a 95% bootstrap confidence interval of [0.18, 1.45].

All performance values are reported as the mean and standard deviation over the 10 random seeds in each training step, ensuring reliable convergence estimates. The interval values of the metrics are given in Appendix B.

Our primary configuration demonstrated meaningful improvements over the baseline, achieving $\approx$ 10.0% higher episode rewards (-67.06 ± 58.18 vs. -76.06 ± 63.42), and up to 40.0% better individual agent rewards (-2.48 ± 2.35 vs. -4.18 ± 4.59), while maintaining comparable policy loss values. These results suggest that the hypergraph communication mechanism helps capture the semantic complexities of the paired observation space.

The results of the ablation analysis conducted across three random seeds suggest that different data preprocessing variants exert only an incremental impact on the performance of the proposed framework (see Appendix C). The addition of Gray Code, while computationally inexpensive, appears to account for most of the observed performance differences by conditioning the latent space of the autoencoder model, whereas input standardization using GPT embeddings seems to have contributed less in this setting. As increased computational complexity can introduce additional operational costs during deployment, the decision to incorporate GPT embeddings therefore needs to be evaluated on a case-by-case basis, which may prove valuable in scenarios involving more diverse or multimodal observation structures.

6. Conclusions and Future Work

This work introduces a neuro-symbolic framework that addresses coordination challenges in multi-agent systems through structured knowledge representations. Two algorithms form the core of the approach: One is a density-adaptive k-NN graph construction method that adjusts neighborhood sizes to local point density while maintaining geometric consistency through Gabriel pruning. The other is a hypergraph aggregation algorithm with tight bounds on the minimum number of bridging hyperedges required under arity and locality constraints.

Empirical validation on PettingZoo's communication-oriented environment demonstrates meaningful performance gains: approximately 10% improvement in episode rewards (-67.06 ± 58.18 vs. -76.06 ± 63.42) and up to 40% improvement in individual agent rewards (-2.48 ± 2.35 vs. -4.18 ± 4.59) compared to DQN baselines, while maintaining comparable policy loss values. The framework successfully handles heterogeneous observations through language model embeddings for format-invariant preprocessing,

while gray encoding regularizes the latent space to ensure semantic coherence across agent knowledge graphs.

The combination of density-adaptive k-NN construction and hypergraph-level arity constraints provides architectural benefits beyond the reported performance improvements. Our graph construction method achieves a lower total reconstruction loss ($4.9 \times 10^{-2} \pm 3.2 \times 10^{-2}$) than that of certain alternatives, including Gabriel graphs ($6.4 \times 10^{-2} \pm 5.0 \times 10^{-2}$), Delaunay Triangulation ($6.2 \times 10^{-2} \pm 3.3 \times 10^{-2}$), and conventional k-NN ($8.0 \times 10^{-2} \pm 5.4 \times 10^{-2}$). Specifically, localized connectivity and bounded hyperedge cardinality mitigate over-smoothing and over-squashing effects—well-documented challenges in deep graph architectures that degrade learning. By maintaining stable message-passing dynamics and improved gradient flow across subgraph boundaries, the approach establishes a foundation for scalable deployment in IoA ecosystems, where interpretability is mandated by regulatory frameworks and communication efficiency directly impacts operational costs [60,61]. The explicit graph structures enable the post hoc auditing of agent interactions and facilitate the integration of domain-specific constraints—capabilities increasingly required for industrial IoA deployments in safety-critical sectors such as smart manufacturing, autonomous logistics, and distributed energy management.

While the proposed algorithms impose no explicit restrictions on graph expansion, the GATv2 performance degrades as the number of hypergraph components increases. This limitation remains manageable for the two-level hierarchical system presented here and can be further alleviated. Thus, future work should investigate sampling-based architectures such as GraphSAGE [62] or minibatch training strategies that maintain scalability without sacrificing representation quality for larger-scale deployments. Beyond the ablation analysis employed, comprehensive evaluation of the framework's robustness across diverse multi-agent scenarios remains an important direction for future work. Additionally, group-theoretic structures, particularly Cayley graphs, offer promising mathematical foundations for high-level agent planning. Such representations could encode action sequences and behavioral patterns as group elements, potentially enabling more structured exploration of the policy space and providing algebraic constraints that guide coordination in complex multi-agent tasks.

The framework's modular design positions it for integration with emerging neuro-symbolic reasoning methods, establishing a path toward more transparent and governable distributed intelligent systems.

Author Contributions: A.I.: writing—original draft, writing—review and editing, and methodology; A.Z.: writing—review and editing, and software; I.T.: writing—review and editing, and software; N.G.: writing—review and editing, methodology, validation, and formal analysis; A.V.: writing—review and editing, investigation, and conceptualization; A.B.: writing—review and editing, resources, funding acquisition, and project administration. All authors have read and agreed to the published version of the manuscript.

Funding: This work supported by the Ministry of Economic Development of the Russian Federation (IGK 000000C313925P4C0002), agreement No 139-15-2025-010.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data presented in this study are available upon reasonable request from the corresponding authors. The data are not publicly available as they involve proprietary information and contractual obligations with collaborating institutions.

Conflicts of Interest: The authors declare no conflicts of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.

Abbreviations

The following abbreviations are used in this manuscript:

AI	Artificial Intelligence;
CTDE	Centralized Training with Decentralized Execution;
CVAE	Conditional Variational Autoencoder;
DDQN	Double Deep Q-Network;
DL	Deep Learning;
DQN	Deep Q-Network;
ELBO	Evidence Lower Bound;
GAE	Graph Autoencoder;
GAT	Graph Attention Network;
HRL	Hierarchical Reinforcement Learning;
IoA	Internet of Agents;
IoT	Internet of Things;
KGE	Knowledge Graph Embedding;
KL	Kullback–Leibler;
LLM	Large Language Model;
MARL	Multi-Agent Reinforcement Learning;
MAS	Multi-Agent System;
MDP	Markov Decision Process;
MLP	Multi-Layer Perceptron;
MPE	Multi-Particle Environment;
MSE	Mean Squared Error;
QMIX	Q-learning Mix Network;
RL	Reinforcement Learning;
SISL	Stanford Intelligent Systems Laboratory;
VAE	Variational Autoencoder.

Appendix A

Appendix A.1. Definitions

Definition A1 (Symmetric k-NN graph)**.** *For $k \in \mathbb{N}$, the undirected symmetric k-NN graph on V_i is $G_k(V_i) = (V_i, E_i)$, where $\{u, v\} \in E_i$ if u is among the k-NN of v in V_i or vice versa.*

Definition A2 (Hypergraph and 2-section)**.** *A hypergraph is a pair $H = (V, E)$ with nonempty finite V and $E \subseteq 2^V \setminus \{\emptyset\}$. Its 2-section $[H]_2$ is the simple graph on V with $\{u, v\} \in [H]_2$ if there exists $e \in E$ such that $\{u, v\} \subseteq e$.*

Definition A3 (Bridging hyperedge)**.** *Let $V = \bigcup_{i=1}^{m} V_i$. A bridging hyperedge is any hyperedge $e \subseteq V$ that intersects vertices from at least two distinct V_i. To keep each $G_k(V_i)$ as a substructure, we include every base edge $\{u, v\} \in E_i$ as a size-2 hyperedge in E; all additional hyperedges we add are bridging hyperedges.*

Definition A4 (Locality radius)**.** *For $\rho > 0$, a bridging hyperedge e has a radius of at most ρ if there exists $x \in \mathbb{R}^2$ such that $\max_{v \in e} \|v - x\| \leq \rho$. This models local radio coverage.*

Definition A5 (Feasibility graph)**.** *For $i \neq j$, we say (i, j) is feasible if there exist $u \in V_i$ and $v \in V_j$ such that either no locality constraint is imposed or $\|u - v\| \leq 2\rho$. The feasibility graph F on vertex set $\{1, \ldots, m\}$ contains edge $\{i, j\}$ if (i, j) is feasible.*

Appendix A.2. Properties

Subgraph retention (P1). For every i, all edges of $G_k(V_i)$ appear as size-2 hyperedges in E. **Cross-graph connectivity (P2).** The 2-section $[H]_2$ is connected. **Bounded arity (P3).** There exists $r \geq 2$ such that every bridging hyperedge has a size of at most r. **Locality (P4).** If enforced, every bridging hyperedge has radius $\leq \rho$. **Load limits (P5).** Each vertex belongs to at most L bridging hyperedges, for some $L \in \mathbb{N}$, to avoid hotspots. **Auditability (P6).** Each bridging hyperedge is labeled with the set of base components it touches, enabling policy checks and revocation.

Appendix A.3. Assumptions

Assumption A1 (Base graphs)**.** *Each $G_k(V_i)$ is a symmetric k-NN graph on $V_i \subset \mathbb{R}^2$ in general position.*

Assumption A2 (Feasible connectivity)**.** *The feasibility graph F is connected. Under locality, this holds, e.g., if for centroids c_i of V_i there exists a spanning tree T of $\{c_i\}$ such that for each $\{i, j\} \in T$ some $u \in V_i$ and $v \in V_j$ satisfy $\|u - v\| \leq 2\rho$.*

Assumption A3 (Arity cap)**.** *Every bridging hyperedge has cardinality of at most $r \geq 2$. If privacy requires a minimum group size $s_{\min}$, we restrict bridging hyperedges to size in $[s_{\min}, r]$; our bounds depend only on r.*

Appendix A.4. Geometry-Aware Refinement

Let β denote the maximum number of base components that can be jointly bridged within a single hyperedge under geometric or semantic constraints ($2 \leq \beta \leq r$). Then, Theorem 1 holds with r replaced with β, i.e., $\lceil (m-1)/(\beta-1) \rceil \leq t^\star \leq m-1$. When locality or feature-space limits restrict $\beta < r$, the lower bound adjusts accordingly, while optimality and tightness (star vs. chain constructions) remain valid. Here, β represents the effective arity determined by the geometry rather than the nominal cap r.

Appendix B

Table A1. Episode reward comparison across training steps.

Learning Interval	Baseline + Comm.	DDQN	DQN
1000–2000	-98.88 ± 75.92	$\mathbf{-93.26 \pm 67.93}$	-162.46 ± 85.18
2000–3000	$\mathbf{-65.30 \pm 65.78}$	-77.764 ± 71.99	-198.44 ± 120.35
3000–4000	$\mathbf{-61.24 \pm 55.33}$	-75.50 ± 67.80	-177.64 ± 105.77
4000–5000	$\mathbf{-64.95 \pm 47.84}$	-77.48 ± 55.41	-163.22 ± 78.32
5000–6000	$\mathbf{-77.33 \pm 62.67}$	-93.93 ± 80.82	-212.40 ± 121.17
6000–7000	$\mathbf{-67.45 \pm 60.55}$	-71.29 ± 61.25	-185.35 ± 105.42
7000–8000	$\mathbf{-49.48 \pm 45.99}$	-65.53 ± 50.00	-146.76 ± 79.20
8000–9000	$\mathbf{-65.64 \pm 59.42}$	-67.66 ± 62.54	-151.46 ± 92.16
9000–10,000	$\mathbf{-53.25 \pm 50.13}$	-62.13 ± 53.05	-160.39 ± 93.73
Overall avg. reward	$\mathbf{-67.06 \pm 58.18}$	-76.06 ± 63.42	-173.12 ± 97.92
Best step over all seeds	$\mathbf{-41.63 \pm 39.96}$	-53.59 ± 49.69	-106.47 ± 52.17

Note: The table presents the values of the mean $\pm$ one standard deviation. The best results among the compared algorithms are in bold.

Table A2. Agent reward comparison across training steps.

Learning Interval	Baseline + Comm.	DDQN	DQN
1000–2000	-9.29 ± 7.35	$\mathbf{-8.64 \pm 6.79}$	-16.61 ± 9.28
2000–3000	$\mathbf{-5.73 \pm 6.76}$	-6.98 ± 7.54	-20.03 ± 11.59
3000–4000	$\mathbf{-5.04 \pm 5.64}$	-6.71 ± 7.22	-19.28 ± 11.38
4000–5000	$\mathbf{-5.05 \pm 4.84}$	-6.62 ± 5.34	-14.43 ± 7.73
5000–6000	$\mathbf{-6.29 \pm 6.54}$	-8.19 ± 7.96	-18.21 ± 10.80
6000–7000	$\mathbf{-5.71 \pm 6.24}$	-6.06 ± 5.59	-18.01 ± 9.86
7000–8000	$\mathbf{-3.82 \pm 4.33}$	-5.11 ± 4.77	-11.88 ± 7.42
8000–9000	$\mathbf{-5.17 \pm 5.31}$	-5.28 ± 6.35	-14.77 ± 9.00
9000–10,000	$\mathbf{-3.77 \pm 4.31}$	-4.98 ± 5.42	-14.91 ± 8.92
Overall avg. reward	$\mathbf{-5.54 \pm 5.70}$	-6.51 ± 6.33	-16.46 ± 9.55
Best step over all seeds	$\mathbf{-2.48 \pm 2.35}$	-4.18 ± 4.59	-10.93 ± 6.67

Note: The table presents the values of the mean $\pm$ one standard deviation. The best results among the compared algorithms are in bold.

Appendix C

Table A3. Evaluation of preprocessing impact.

Metric	Gray Code + GPT	Gray Code	GPT	Without
Episode reward	$\mathbf{-66.02 \pm 61.34}$	-66.47 ± 63.18	-68.73 ± 65.92	-68.95 ± 67.45
Agent reward	$\mathbf{-5.25 \pm 5.11}$	-5.37 ± 5.26	-5.73 ± 5.49	-5.76 ± 5.62

Note: The table presents the values of the mean $\pm$ one standard deviation for four preprocessing pipelines. The pipelines compare various combinations of Gray Code and GPT embeddings, as well as a baseline with no preprocessing beyond zero-padding to equal length. The best results among the compared algorithms are in bold.

References

1. Wang, Y.; Guo, S.; Pan, Y.; Su, Z.; Chen, F.; Luan, T.H.; Li, P.; Kang, J.; Niyato, D. Internet of Agents: Fundamentals, Applications, and Challenges. *arXiv* **2025**, arXiv:2505.07176. [CrossRef]
2. Chen, W.; You, Z.; Li, R.; Guan, Y.; Qian, C.; Zhao, C.; Yang, C.; Xie, R.; Liu, Z.; Sun, M. Internet of Agents: Weaving a Web of Heterogeneous Agents for Collaborative Intelligence. In Proceedings of the 2025 International Conference on Learning Representations (ICLR), Singapore, 24–28 April 2025.
3. Luzolo, P.H.; Elrawashdeh, Z.; Tchappi, I.; Galland, S.; Outay, F. Combining multi-agent systems and artificial intelligence of things: Technical challenges and gains. *Internet Things* **2024**, *28*, 101364. [CrossRef]
4. Mandaric, K.; Keselj Dilberovic, A.; Ježic, G. A multi-agent system for service provisioning in an Internet-of-Things smart space based on user preferences. *Sensors* **2024**, *24*, 1764. [CrossRef] [PubMed]
5. Wang, X.; He, P.; Yao, H.; Shi, X.; Wang, J.; Guo, Y. A Dual Fusion Pipeline to Discover Tactical Knowledge Guided by Implicit Graph Representation Learning. *Mathematics* **2024**, *12*, 528. [CrossRef]
6. Liu, S.; Marris, L.; Lanctot, M.; Piliouras, G.; Leibo, J.Z.; Heess, N. Neural Population Learning beyond Symmetric Zero-Sum Games. In Proceedings of the 23rd International Conference on Autonomous Agents and Multiagent Systems (AAMAS'24), Auckland, New Zealand, 6–10 May 2024; International Foundation for Autonomous Agents and Multiagent Systems: Richland, SC, USA, 2024; pp. 1247–1255.
7. Littman, M.L. Markov games as a framework for multi-agent reinforcement learning. In *Machine Learning Proceedings*; Morgan Kaufmann: San Francisco, CA, USA, 1994; pp. 157–163.
8. Zhang, K.; Yang, Z.; Başar, T. Multi-agent reinforcement learning: A selective overview of theories and algorithms. In *Handbook of Reinforcement Learning and Control*; Springer: Cham, Switzerland, 2021; pp. 321–384.
9. Jin, W.; Du, H.; Zhao, B.; Tian, X.; Shi, B.; Yang, G. A comprehensive survey on multi-agent cooperative decision-making: Scenarios, approaches, challenges and perspectives. *arXiv* **2025**, arXiv:2503.13415.
10. Gronauer, S.; Diepold, K. Multi-agent deep reinforcement learning: A survey. *Artif. Intell. Rev.* **2022**, *55*, 895–943. [CrossRef]
11. Ning, Z.; Xie, L. A survey on multi-agent reinforcement learning and its application. *J. Autom. Intell.* **2024**, *3*, 73–91. [CrossRef]
12. Iqbal, S.; Sha, F. Actor-attention-critic for multi-agent reinforcement learning. In Proceedings of the International Conference on Machine Learning (ICML), Long Beach, CA, USA, 9–15 June 2019.

13. Christianos, F.; Schäfer, L.; Albrecht, S.V. Shared experience actor-critic for multi-agent reinforcement learning. In Proceedings of the 31st International Conference on Neural Information Processing Systems, Long Beach, CA, USA, 4–9 December 2019; Volume 33, pp. 10707–10717.
14. Lowe, R.; Wu, Y.; Tamar, A.; Harb, J.; Abbeel, P.; Mordatch, I. Multi-agent actor-critic for mixed cooperative–competitive environments. In Proceedings of the Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, Long Beach, CA, USA, 4–9 June 2017.
15. Sukhbaatar, S.; Szlam, A.; Fergus, R. Learning multiagent communication with backpropagation. In Proceedings of the Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, Barcelona, Spain, 5–10 December 2016.
16. Foerster, J.N.; Assael, Y.M.; de Freitas, N.; Whiteson, S. Learning to communicate with deep multi-agent reinforcement learning. In Proceedings of the Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, Barcelona, Spain, 5–10 December 2016.
17. Das, A.; Gervet, T.; Romoff, J.; Batra, D.; Parikh, D.; Rabbat, M.; Pineau, J. TarMAC: Targeted multi-agent communication. In Proceedings of the International Conference on Machine Learning (ICML), Long Beach, CA, USA, 9–15 June 2019.
18. Hu, S.; Shen, L.; Zhang, Y.; Tao, D. Learning multi-agent communication from a graph modeling perspective (CommFormer). In Proceedings of the 2024 International Conference on Learning Representations (ICLR), Vienna, Austria, 7–11 May 2024.
19. Niu, Y.; Paleja, R.; Gombolay, M. Multi-agent graph-attention communication and teaming. In Proceedings of the 2021 International Conference on Autonomous Agents and Multiagent Systems (AAMAS), Virtual, 3–7 May 2021.
20. Qin, Z.; Lu, Y. Knowledge graph-enhanced multi-agent reinforcement learning for adaptive scheduling in smart manufacturing. *J. Intell. Manuf.* **2024**, *36*, 5943–5966. [CrossRef]
21. Amato, A.; Galliera, R.; Venable, K.B.; Suri, N. Encoding goals as graphs: Structured objectives for scalable cooperative multi-agent reinforcement learning. In Proceedings of the 2025 CoCoMARL Workshop, Edmonton, AB, Canada, 5 August 2025.
22. Li, C.; Liu, J.; Zhang, Y.; Wei, Y.; Niu, Y.; Yang, Y.; Liu, Y.; Ouyang, W. ACE: Cooperative multi-agent Q-learning with bidirectional action-dependency. In Proceedings of the 2023 AAAI Conference on Artificial Intelligence, Washington DC, USA, 7–14 February 2023.
23. Kontogiannis, A.; Papathanasiou, K.; Shen, Y.; Stamou, G.; Zavlanos, M.M.; Vouros, G. Enhancing cooperative multi-agent reinforcement learning with state modelling and adversarial exploration. *arXiv* **2025**, arXiv:2505.05262. [CrossRef]
24. Guestrin, C.; Lagoudakis, M.; Parr, R. Coordinated reinforcement learning. In Proceedings of the 2002 International Conference on Machine Learning (ICML), Helsinki, Finland, 19–23 August 2002.
25. Böhmer, W.; Kurin, V.; Whiteson, S. Deep coordination graphs. In Proceedings of the 2020 International Conference on Machine Learning (ICML), Vienna, Austria, 12–18 July 2020.
26. DeLong, L.N.; Mir, R.F.; Fleuriot, J.D. Neurosymbolic AI for reasoning over knowledge graphs: A survey. *IEEE Trans. Neural Netw. Learn. Syst.* **2024**, *36*, 7822–7842. [CrossRef] [PubMed]
27. Hitzler, P.; Sarker, M.K. Neuro-symbolic artificial intelligence: The state of the art. In *Neuro-Symbolic Artificial Intelligence: The State of the Art*; Frontiers in Artificial Intelligence and Applications Series; IOS Press: Amsterdam, The Netherlands, 2021; Volume 342.
28. Zhong, L.; Wu, J.; Li, Q.; Peng, H.; Wu, X. A comprehensive survey on automatic knowledge graph construction. *ACM Comput. Surv.* **2024**, *56*, 94. [CrossRef]
29. Chen, Z.; Zhang, Y.; Fang, Y.; Geng, Y.; Guo, L.; Chen, X.; Li, Q.; Zhang, W.; Chen, J.; Zhu, Y.; et al. Knowledge graphs meet multi-modal learning: A comprehensive survey. *arXiv* **2024**, arXiv:2402.05391. [CrossRef]
30. Zhang, Y.; Yang, Q.; An, D.; Zhang, C. Coordination between individual agents in multi-agent reinforcement learning. In Proceedings of the AAAI Conference on Artificial Intelligence, Virtual, 2–9 February 2021; Volume 35, pp. 11387–11394.
31. Rashid, T.; Samvelyan, M.; Schroeder de Witt, C.; Farquhar, G.; Foerster, J.; Whiteson, S. QMIX: Monotonic value function factorisation for deep multi-agent reinforcement learning. In Proceedings of the 2018 International Conference on Machine Learning (ICML), Stockholm, Sweden, 10–15 July 2018.
32. Terry, J.; Black, B.; Grammel, N.; Jayakumar, M.; Hari, A.; Sullivan, R.; Santos, L.S.; Dieffendahl, C.; Horsch, C.; Perez-Vicente, R.; et al. PettingZoo: A standard API for multi-agent reinforcement learning. *arXiv* **2021**, arXiv:2009.14471.
33. Li, Q.; Han, Z.; Wu, X.-M. Deeper Insights into Graph Convolutional Networks for Semi-Supervised Learning. In Proceedings of the AAAI Conference on Artificial Intelligence, New Orleans, LA, USA, 2–7 February 2018; Volume 32, pp. 3538–3545.
34. Alon, U.; Yahav, E. On the Bottleneck of Graph Neural Networks and its Practical Implications. In Proceedings of the 9th International Conference on Learning Representations (ICLR), Virtual, 3–7 May 2021.
35. Topping, J.; Di Giovanni, F.; Chamberlain, B.P.; Dong, X.; Bronstein, M.M. Understanding Over-Squashing and Bottlenecks on Graphs via Curvature. In Proceedings of the 10th International Conference on Learning Representations (ICLR), Virtual, 25–29 April 2022.
36. Sheng, J.; Wang, X.; Jin, B.; Yan, J.; Li, W.; Chang, T.-H.; Wang, J.; Zha, H. Learning structured communication for multi-agent reinforcement learning. *arXiv* **2020**, arXiv:2002.04235. [CrossRef]

37. Veličković, P.; Cucurull, G.; Casanova, A.; Romero, A.; Liò, P.; Bengio, Y. Graph attention networks. In Proceedings of the 2018 International Conference on Learning Representations (ICLR), Vancouver, BC, Canada, 30 April–3 May 2018.
38. Brody, S.; Alon, U.; Yahav, E. How attentive are graph attention networks? (GATv2). In Proceedings of the 2021 International Conference on Learning Representations (ICLR), Virtual, 3–7 May 2021.
39. Lee, D.T.; Schachter, B.J. Two algorithms for constructing a Delaunay triangulation. *Int. J. Comput. Inf. Sci.* **1980**, *9*, 219–242. [CrossRef]
40. Gabriel, K.R.; Sokal, R.R. A new statistical approach to geographic variation analysis. *Syst. Zool.* **1969**, *18*, 259–278. [CrossRef]
41. Mao, J.; Tenenbaum, J.B.; Wu, J. Neuro-symbolic concepts. *arXiv* **2025**, arXiv:2505.06191. [CrossRef]
42. Rivas, A.; Collarana, D.; Torrente, M.; Vidal, M.-E. A neuro-symbolic system over knowledge graphs for link prediction. *Semant. Web J.* **2023**, *15*, 1–25. [CrossRef]
43. Chen, S.; Cai, Y.; Fang, H.; Huang, X.; Sun, M. Differentiable neuro-symbolic reasoning on large-scale knowledge graphs. In Proceedings of the 37th Conference on Neural Information Processing Systems NeurIPS 2023, New Orleans, LA, USA, 10–16 December 2023.
44. Li, Z.; Ao, Y.; He, J. SpherE: Expressive and interpretable knowledge graph embedding for set retrieval. In Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI), Guangzhou, China, 29–31 August 2025.
45. Zhang, Z.; Cai, J.; Zhang, Y.; Wang, J. Learning hierarchy-aware knowledge graph embeddings for link prediction. In Proceedings of the AAAI Conference on Artificial Intelligence, New York, NY, USA, 7–12 February 2020; Volume 34, pp. 3065–3072.
46. Ortega-Guzmán, V.H.; Gutiérrez-Preciado, L.; Cervantes, F.; Alcaraz-Mejia, M. A Methodology for Knowledge Discovery in Labeled and Heterogeneous Graphs. *Appl. Sci.* **2024**, *14*, 838. [CrossRef]
47. Feng, Q.; Zhao, T.; Liu, C. A "Pipeline"-Based Approach for Automated Construction of Geoscience Knowledge Graphs. *Minerals* **2024**, *14*, 1296. [CrossRef]
48. Dadman, S.; Bremdal, B.A. Crafting Creative Melodies: A User-Centric Approach for Symbolic Music Generation. *Electronics* **2024**, *13*, 1116. [CrossRef]
49. Li, I.; Fabbri, A.R.; Hingmire, S.; Radev, D. R-VGAE: Relational-variational graph autoencoder for unsupervised prerequisite chain learning. *Knowl. Based Syst.* **2021**, *228*, 107291.
50. Liu, J.; Mao, Q.; Lin, C.; Song, Y.; Li, J. LatentLogic: Learning logic rules in latent space over knowledge graphs. In Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP), Singapore, 6–10 December 2023; pp. 4578–4586.
51. Xia, R.; Zhang, Y.; Zhang, C.; Liu, X.; Yang, B. Multi-head variational graph autoencoder constrained by sum-product networks. In Proceedings of the Web Conference (WWW), Austin, TX, USA, 30 April–4 May 2023; pp. 641–650.
52. Kingma, D.P.; Welling, M. Auto-encoding variational Bayes. *arXiv* **2013**, arXiv:1312.6114.
53. Rezende, D.J.; Mohamed, S.; Wierstra, D. Stochastic backpropagation and approximate inference in deep generative models. In Proceedings of the International Conference on Machine Learning (ICML), Beijing, China, 21–26 June 2014.
54. Sohn, K.; Lee, H.; Yan, X. Learning structured output representation using deep conditional generative models. In Proceedings of the 29th International Conference on Neural Information Processing Systems, Montreal, QC, Canada, 7–12 December 2015; Volume 28.
55. Edelsbrunner, H. *Geometry and Topology for Mesh Generation*; Cambridge University Press: Cambridge, UK, 1987.
56. Jaromczyk, J.W.; Toussaint, G.T. Relative neighborhood graphs and their relatives. *Proc. IEEE* **1992**, *80*, 1502–1517. [CrossRef]
57. Brito, M.; Yukich, J.E. Connectivity of the nearest neighbor graph of random points in convex sets. *Stat. Probab. Lett.* **1997**, *38*, 301–306.
58. Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A.A.; Veness, J.; Bellemare, M.G.; Graves, A.; Riedmiller, M.; Fidjeland, A.K.; Ostrovski, G. Human-level control through deep reinforcement learning. *Nature* **2015**, *518*, 529–533. [CrossRef] [PubMed]
59. Ozaki, Y.; Watanabe, S.; Yanase, T. OptunaHub: A Platform for Black-Box Optimization. *arXiv* **2025**, arXiv:2510.02798.
60. Alsboui, T.; Qin, Y.; Hill, R.; Al-Aqrabi, H. Distributed Intelligence in the Internet of Things: Challenges and Opportunities. *SN Comput. Sci.* **2021**, *2*, 277. [CrossRef]
61. Bhuyan, B.P.; Jaiswal, V.; Cherif, A.R. A Knowledge Representation System for the Indian Stock Market. *Computers* **2023**, *12*, 90. [CrossRef]
62. Hamilton, W.; Ying, R.; Leskovec, J. Inductive representation learning on large graphs. In Proceedings of the Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, Long Beach, CA, USA, 4–9 June 2017; Volume 30, pp. 1024–1034.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.