LEARNING OBJECT-CENTERED AUTOTELIC BEHAVIORS WITH GRAPH NEURAL NETWORKS

Ahmed Akakzia
Sorbonne Université
ahmed.akakzia@isir.upmc.fr

Olivier Sigaud
Sorbonne Université

ABSTRACT

Although humans live in an open-ended world and endlessly face new challenges, they do not have to learn from scratch each time they face the next one. Rather, they have access to a handful of previously learned skills, which they rapidly adapt to new situations. In artificial intelligence, autotelic agents — which are intrinsically motivated to represent and set their own goals — exhibit promising skill adaptation capabilities. However, these capabilities are highly constrained by their policy and goal space representations. In this paper, we propose to investigate the impact of these representations on the learning capabilities of autotelic agents. We study different implementations of autotelic agents using four types of Graph Neural Networks policy representations and two types of goal spaces, either geometric or predicate-based. We show that combining object-centered architectures that are expressive enough with semantic relational goals enables an efficient transfer between skills and promotes behavioral diversity. We also release our graph-based implementations to encourage further research in this direction.

1 INTRODUCTION

A central challenge in artificial intelligence (AI) consists in designing artificial agents capable of solving an unrestricted set of tasks in a continual and open-ended skill learning process. In principle, these processes should be domain-agnostic. Reinforcement learning (RL) seems to be an adequate paradigm to solve a single sequential decision problem from a reward signal (Sutton et al., 1999). Nevertheless, this signal is usually predetermined and highly grounded to its designer's aspirations. Thus, the extension of the RL framework to an open-ended sequences of unpredictable tasks raises difficult questions.

Recently, a promising line of research has been interested in the design of *autotelic agents*, borrowing older ideas from (Steels, 2004). These agents are intrinsically motivated to represent, set and pursue their own goals. Usually, they do not depend on any external reinforcement signal, since they autonomously reward themselves over the completion of their own goals. Autotelic agents are known to be open-ended learners. Through RL, they manage to acquire goal-directed behaviors which can transfer to domains sharing similar goal spaces. However, this transfer is deeply bound to their representational capabilities.

From that perspective, a key challenge consists in endowing autotelic agents with appropriate inductive biases to enhance their representational power. To enable efficient transfer, such biases should express a set of general and structured features. On the one hand, the design of the autotelic agents' goal spaces should depend on some abstract general rules rather than geometric properties specific to a particular domain. Namely, recent works in AI (Akakzia et al., 2021; Alomari et al., 2017; Kulick et al., 2013; Tellex et al., 2011) introduced symbolic high-level object-centered representations to explicitly capture abstract spatial relations such as proximity and aboveness, which we used to refer to the quality of being directly above. By contrast, other works use plain spatial target coordinates specific to each of the available objects (Colas et al., 2019; Li et al., 2019; Lanier et al., 2019).

On the other hand, although neural networks are flexible tools to learn latent representations, their raw usage is insufficient to capture disentangled representations from high-dimensional structured input. Recently, Graph Neural Networks (GNNs) have been introduced to implement relational inductive biases in neural networks. They mainly rely on shared networks to transfer features among the input components. Besides, they follow efficient computation schemes: through their neighborhood aggregation and graph-level pooling schemes, they easily capture the existing relationships between nodes.

Contributions. In this paper, we study the use of GNNs in autotelic learning within a multi-object manipulation domain. More specifically, we investigate 4 variants of GNNs: *full graph networks*, *interaction networks*, *relation networks* and *deep sets*. Furthermore, we consider two different types of goal spaces: 1) *semantic goals* based on binary predicates describing spatial relations between physical objects; 2) *continuous goals* corresponding to specific target positions for each object. For multi-object manipulation domains, we show that:

- Compared to flat architectures which directly leverage entangled stream of input features, graph-based neural networks are better suited.
- Full graph networks and interaction networks outperform the other GNN-based architectures in learning actionable representations.
- Coupling graph-based architectures with semantic goals helps to efficiently transfer between goals and further improves the behavioral diversity.

Finally, we release our implementations of the considered GNN-based architectures in multi-object manipulation domain to encourage further research in this direction[1]

2 Related work

This paper relies on several previous works from different areas of research within AI. Namely, we consider recent findings in automatic curriculum learning, semantic goal representations, graph neural networks and graph-based autotelic learning.

Automatic Curriculum Learning. Adaptability is a key characteristic enabling humans to display an exceptional capacity to learn (Elman, 1993) and works in AI attempted to leverage similar automatic curriculum learning (ACL) schemes in artificial agents (Portelas et al., 2020). Most of these approaches leverage forms of intrinsic motivations to power their exploration and learning progress (LP) (Bellemare et al., 2016; Achiam & Sastry, 2017; Nair et al., 2018; Burda et al., 2018; Pathak et al., 2019; Colas et al., 2019; Pong et al., 2019). In this paper, we borrow the LP-based curriculum learning algorithm introduced in Colas et al. (2019).

Semantic Goal Representations. Studies in developmental psychology suggest that notions such as proximity, animacy and containment are innately grounded in the perceptual world of the infant (Mandler, 2012). Inspired by this line of thought, recent works in AI introduced symbolic high-level representations to explicitly capture abstract spatial relations (Tellex et al., 2011; Kulick et al., 2013; Alomari et al., 2017; Akakzia et al., 2021). We borrow the semantic goal representations used in Akakzia et al. (2021) and based on the predicates *close* and *above*.

Graph Neural Networks. GNNs are powerful tools to implement strong inductive biases that focus on structured representations (Battaglia et al., 2018). At the price of more computations, they efficiently foster combinatorial generalization and improve sample efficiency over standard architectures in different machine learning domains (Gilmer et al., 2017; Scarselli et al., 2005; Zaheer et al., 2017; Li et al., 2019). GNNs parse the stream of input features into several objects, called *nodes*. They also capture the relational features between pairs of these objects which they store in the corresponding *edges*. They usually involve three computational schemes: 1) **Edge updates** using the initial features of the edge and both features of the nodes involved within that edge; 2) **Node updates** using the initial features of the node and the aggregated features of the edges that

[1]https://github.com/akakzia/rlgraph.

enter that nodes; 3) **Graph output** using an aggregation of either all the nodes or the edges features. The first two steps involve shared networks, which enable transfer between the different nodes and edges. Depending on the order and the nature of the computational steps, there exist many variants of GNNs. In this paper, we only consider 4 of these variants: *full graph networks* (Battaglia et al., 2018), *interaction networks* (Battaglia et al., 2016), *relation networks* (Santoro et al., 2017) and *deep sets* (Zaheer et al., 2017). Details about the implementations of these variants are provided in Section 3.2.

Graph-based Autotelic RL. GNNs have been used to solve RL problems (Zambaldi et al., 2018; Li et al., 2019; Colas et al., 2020; Akakzia et al., 2021). By contrast to Li et al. (2019); Colas et al. (2020); Akakzia et al. (2021) — which explicitly associate a node to each object in an object manipulation domain — the approach in Zambaldi et al. (2018) attempts to solve the StarCraft II mini-games (Vinyals et al., 2017) without object-centered inductive bias. In the latter, the nodes do not correspond to specific objects, but rather to randomly scattered boxes of pixels. In this paper, we rather join the former group.

3 METHODS

In this section, we first introduce the object manipulation environment and the two goal spaces we use in this paper (Section 3.1). Then, we present the graph-based implementations of our autotelic agents (Section 3.2)

3.1 ENVIRONMENT AND GOAL SPACES

The Fetch Manipulate Environment. Agents evolve in the *Fetch Manipulate* domain from Akakzia et al. (2020), which is a variant of the standard *Fetch* domains (Plappert et al., 2018). We extend it to a 5-object setup: the agent is a 4-DoF robotic arm facing 5 colored objects on a table. It perceives features of its body and of the surrounding objects. These features include geometric positions, orientations and velocities.

Autotelic Learning with Semantic Goals. We consider high-level binary representations that assert the presence (1) or absence (0) of the binary spatial relations *above* and *close* between objects. As the latter is symmetric (*close(A, B) = close(B, A)*), we only consider 10 combinations of objects for this predicate. However, we consider all the 20 ordered pairs of objects for the *above* predicate. This yields semantic goal vectors of 30 dimensions. The resulting configuration space contains 2^{30} elements, among which ~ 75.000 are physically reachable. These semantic representations are inspired by the work of Mandler (2012) on a minimal set of spatial primitives children seem to be born with, or to develop early in life. Initially empty, the set of discovered semantic goals gets gradually filled each time an agent encounters new configurations. Accordingly, agents autonomously select and attempt to master goals from this set. They reward themselves for each correctly placed object (i.e. all the predicates involving that object are verified). An episode ends successfully if all objects are placed correctly before a time limit. At the beginning of an episode, the blocks are procedurally placed on the table so that they can never be initially stacked.

Autotelic Learning with Continuous Goals. Continuous goals correspond to precise target positions for each available object. To succeed, agents have to place every object in its corresponding target position. See Figure 1 for an illustration. These goal spaces are used in many works attempting to solve multi-object manipulation problems (Colas et al., 2019; Li et al., 2019; Lanier et al., 2019). We suppose that these agents are initially aware that they can construct stacks using the available objects, and that the maximum number of objects stacked corresponds to the number of available objects. At the beginning of each episode, agents autonomously select how many objects they want to stack (from 0 up to 5 in this paper). Accordingly, target positions are generated for each object. Agents reward themselves for each object placed correctly within a range of its corresponding target position. An episode ends successfully if all objects are placed correctly before a time limit. To further accelerate the learning process, we consider biased initializations as part of a way to adapt the difficulty of the task to the learner's skills: at the beginning of each

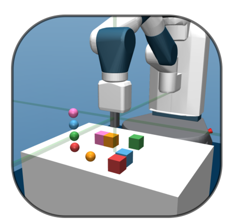


Figure 1: Illustration of objects and targets.

episode, and with a probability of 0.2, blocks are arranged into a stack of up to 5 objects. We can view this ZPD *management* as this accounts for Vygotski's notion of *Zone of Proximal Development* Vygotsky (1978). To stabilize the learning process, we use an automatic LP-based curriculum (Colas et al., 2019): based on their learning progress estimations, agents can choose to target goals with no stacks, a stack of 2, 3, 4 or 5 objects where all target positions that are not involved in stacks are automatically generated directly on the table. See Appendix B.1 for more details about the importance of ACL when using continuous goals.

3.2 GRAPH-BASED AUTOTELIC LEARNING

The agents we consider in this paper are autotelic. In this section, we describe the implementation of the intrinsically motivated goal-conditioned RL module. It is powered by the Soft-Actor Critic algorithm (SAC) (Haarnoja et al., 2018). We choose to model both the critic and the policy based on GNNs. We use the Multi-criteria Hindsight Experience Replay algorithm (MC-HER) to facilitate transfer between goals (Lanier et al., 2019). MC-HER extends the Hindsight Experience Replay (HER) strategy to multi-object scenarios. As the latter focuses on the whole scene when assigning fictive rewards to future achieved goals, the former makes use of the incremental property of per-object rewards to enable further transfer between partial features of the goal vector.

3.2.1 GRAPH STRUCTURE

All our agents use a fully connected graph structure: every object corresponds to a node, and all nodes are connected. First, each node holds the features of a particular object in the scene. Second, each edge linking a source and a recipient node holds partial features of the goal. For semantic goals, these features correspond to the predicates that involve both the source and the recipient node, while for continuous goals, they correspond to the target position of the block corresponding to the source node. Finally, the global features correspond either to the agent's body state (in the case of the policy) or to a concatenation of the agent's body state and the action (in the case of the critic). We respectively denote the node features, edges features and global features by X, E and U.

3.2.2 GRAPH COMPUTATIONS

Although all our agents rely on the same graph structure, they use different computation schemes. In this paper, we focus on four particular types of GNNs: full graph networks (GN), interaction networks (IN), relation networks (RN) and deep sets (DS). Figure 2 illustrates the different computation steps for each architecture.

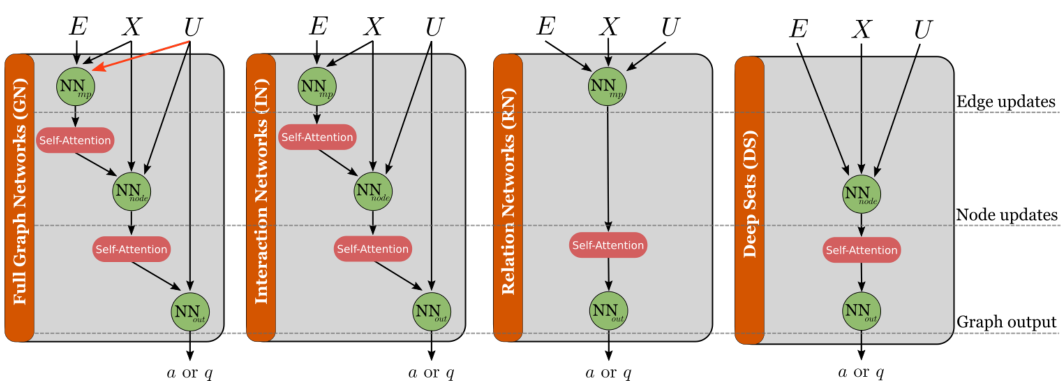


Figure 2: Illustration of the different computational schemes for (from left to right) GN, IN, RN and DS. E, X and U respectively correspond to the edge features, node features and global features. Note that GN uses U to update edges features (red arrow), while IN does not and RN only updates edges features, while DS only updates nodes features.

Full Graph Network (GN). As its name suggests, this architecture uses the whole computation scheme within a standard graph network block. See Figure 2 for an illustration. First, an *edge update* step is performed. A shared network NN_{mp} is used to compute the update features of each edge. It takes as input the concatenated input features of each edge (goal features), the involved source and recipient nodes (object features) and the global features. Second, a *node update* step is performed for each node using a second shared network NN_{node}. It takes as input the concatenated input features of the considered node, the global features and an aggregation of the updated features of the incoming edges. Third, the *graph output* step is performed, where the updated features of the nodes are pooled, concatenated with the global features and fed to a readout network NN_{out}. The output quantity corresponds to either the action (in the case of the actor) or the q-value (in the case of the critic). In this paper, we use self-attention to compute the weighing scores used in all the aggregation steps (Vaswani et al., 2017; Veličković et al., 2017).

Interaction Network (IN). This architecture resembles the one described in the GN architecture. The only difference is that, during the edge update step, the global features are not used as inputs to the shared network NN_{mp}.

Relation Network (RN). This architecture entirely bypasses the node update step. It only performs the edge update step using the shared network NN_{mp}, which takes as inputs the initial node, edge and global features. The output vector is aggregated using a self-attention module, then fed to a readout network NN_{out}.

Deep Sets (DS). This architecture entirely bypasses the edge update step. It only performs node updates using the shared network NN_{node}. The latter takes as input the node, edge and global features, outputs a vector is later fed to a self-attention module to compute attention scores. Finally, the aggregated vector is fed to a readout network NN_{out}.

4 Experiments and results

We train 4 graph-based autotelic agents in the *Fetch Manipulate* domain with 5 objects using the graph architectures described in Section 3.2. We consider both the semantic and continuous goal spaces introduced in Section 3.1.

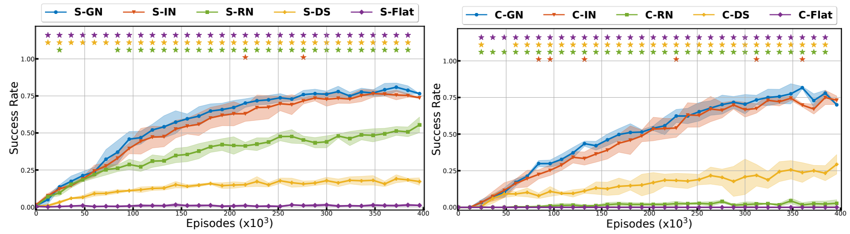
Evaluation Classes. To evaluate the agents, we define several evaluation classes for both semantic and continuous goals. First, for semantic goals, we consider classes of configurations where exactly i pairs of blocks are close (C_i), configurations containing stacks of size i (S_i), configurations containing pyramids of size 3 (P_3) and combinations of these. These classes are disjoint and their union does not cover the entire semantic configuration space, but they are representative enough and they enable fair comparisons between the agents. Second, for continuous goals, we consider classes of configurations where there are no stacks and where there is a stack of i objects ($\widetilde{S}_i$, where the symbol $\sim$ is for continuous).

Evaluation Metrics. Evaluations are performed each 50 cycles. During one cycle, the agents perform 2 rollouts of 200 timesteps with 2 goals sampled autonomously. At test time, the per-class performance of the agent is computed on 24 goals of each evaluation class (264 semantic goals and 120 continuous goals). The measure of the agent's global success rate (SR) is the average of all the per-class successes. Testing is conducted offline and with deterministic policies.

Baseline. For both semantic and continuous goals, we consider a flat baseline, where all the perceived features are concatenated and directly fed to the neural networks. We call Semantic-Flat (S-FLAT) and Continuous-Flat (C-FLAT) the flat baseline using respectively semantic and continuous goals.

4.1 Global performance metrics

In this section, we study the global performance of the different graph-based autotelic agents. Figure 3 presents the average SR across evaluation classes for both semantic goals (Figure 3a) and continuous goals (Figure 3b).



(a) (b)

Figure 3: Global SR across training episodes with (a) Semantic (S) goals and (b) Continuous (C) goals for the considered agents. Mean $\pm$ standard deviations are computed over 5 seeds. Stars highlight statistical differences w.r.t S-GN agents (Welch's t-test with null hypothesis $\mathcal{H}_0$: no difference in the means, $\alpha = 0.05$).

Semantic Goals. On the one hand, we note that the Semantic-FLAT (S-FLAT) baseline fails to increase its global SR during all training episodes. This suggests that using flat architectures is not suitable for multi-object domains with semantic goals. In fact, neural networks are unable to disentangle the learned features when the input is a raw concatenation of all the perceived states. This most likely becomes intractable in high-dimensional scenarios when the number of objects involved in the sensorimotor interactions increases. On the other hand, all the agents that use object-centered architectures are able to increase their global SR. First, Semantic-DS (S-DS) agents — which bypass the edge update during the edge update step — get stuck at 20% of the maximum global SR. This suggests that, when using semantic relational goals, deep sets do not leverage enough representational power to learn object-centered representations, as they only rely on node updates. Second, the Semantic-RN (S-RN) agents yield better performance. In fact, with semantic relational goals, relation networks are able to pass information about the different predicates during the edge update step. However, bypassing the node update prevents them from maximizing the global SR, as they get stuck at around 50%. Third, both the Semantic-GN (S-GN) and Semantic-IN (S-IN) agents are able to outperform all the other counterparts. In fact, using both the edge and the node updates allows them to gain more representational power. Besides, their performance is similar across all training episodes since the statistical differences only appear rarely (see red stars on Figure 3a). This suggests that using the global features during the edge update step is not necessary.

Continuous Goals. Similar to semantic goals, the Continuous-FLAT (C-FLAT) baseline fails to learn any interesting behavior. However, unlike with semantic goals, not all the graph-based agents are able to increase their global SR across training episodes. In fact, the Continuous-RN (C-RN) agents — which exclusively rely on edge updates during the edge update step — perform in par of the C-FLAT baseline. This shows that, when dealing with geometric goals that do not exhibit relational features, bypassing the node update step raises a red flag: aggregated pairwise node features do not capture actionable information about the relational structure of the objects. Moving on, the global SR of the Continuous-DS (C-DS) agents ,— which exclusively rely on node updates — increases but gets stuck at 25% of the maximum global performance. This not only proves the importance of the node update step, but also suggests that this step alone is not sufficient to maximize the global SR. Furthermore, the Continuous-GN (C-GN) and Continuous-IN (C-IN) agents outperform all the other agents. This further proves the importance of combining both the edge and the node update steps. Besides, they both show similar performance, as statistical differences only occur rarely (see red stars in Figure 3b). This implies that using the global features during the edge update step does not seem to be necessary.

4.1.1 Per Class Performance Metrics

The global performance metrics in Section 4.1 show that the average SR across evaluation classes gets stuck at around 75% for both semantic and continuous agents. To investigate this, we zoom on the per class performance metrics.

Semantic Goals. Figure 4 shows the per-class performance of S-GN, S-IN, S-RN and S-DS. First, and as the global performance metrics suggest (Section 4.1), S-GN and S-IN show very similar local

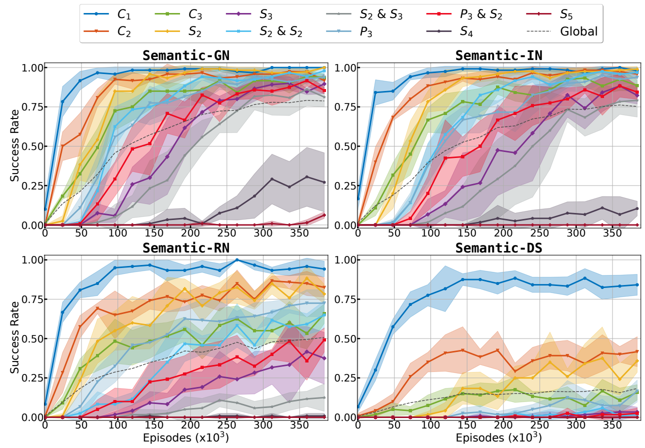


Figure 4: Local SR for each class across training episodes with continuous goals. Mean $\pm$ standard deviations are computed over 5 seeds.

performance. They are both able to master all the classes except for S_4 and S_5. This failure occurs because the learned policies are sub-optimal. In fact, when rewarding themselves for each object placed correctly, the critics would most likely be *greedy*: incremental rewards should come fast, even if this means not constructing stacks in the trivial order (from base upwards). As a result, agents would start by constructing the upper part of a stack, then placing is on the base object. This is not a problem for S_3 since robotic arms can pick and place a stack of two blocks. However, in S_4, it's impossible to pick and place a stack of three blocks. See Figure 5 for an illustrative example. Second, the S-RN agents do not transfer between goals as well as the S-GN and S-IN agents. In fact, the slopes of the curves in S-RN are smaller than the ones in S-GN and S-IN. It seems that S-RN in unable to efficiently learn about many classes at the same time. Finally, as suggested by the global performance metrics in Section 4.1, the deep sets architecture used in S-DS agents does not have enough representational power to learn stacks.

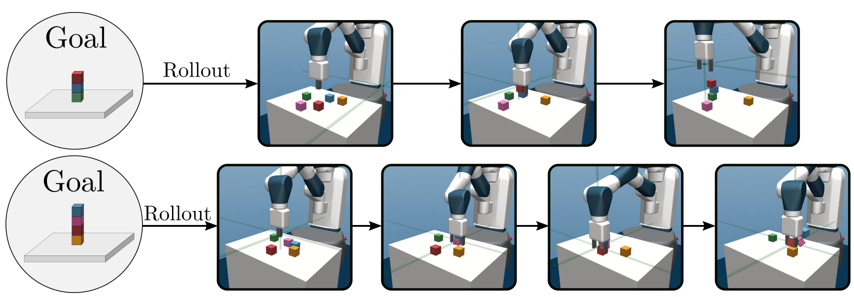


Figure 5: Example of sub-optimal behavior with semantic goals when targeting a goal in S_3 (up) and in S_4 (down). The agent tries to pick and place a stack of three objects and fails (down).

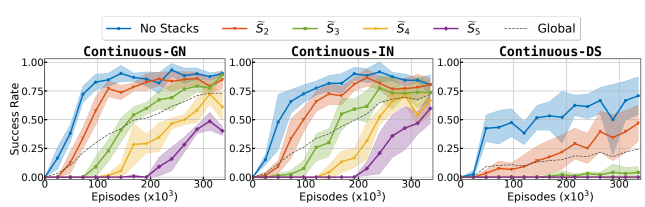
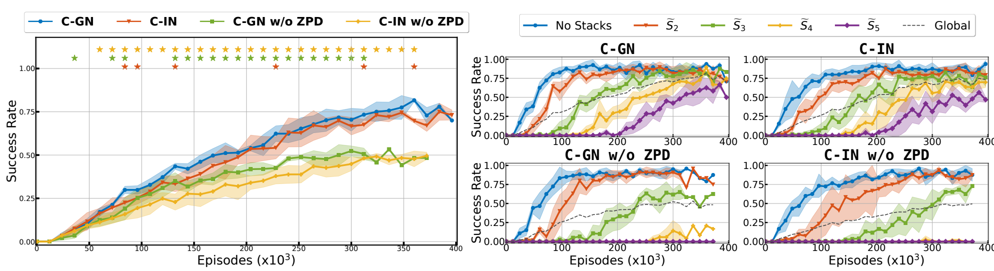


Figure 6: Local SR for each class across training episodes with continuous goals. Mean $\pm$ standard deviations are computed over 5 seeds.

Continuous Goals. Figure 6 shows the per-class performance of C-GN, C-IN and C-DS. All the agents first start mastering the easy classes, before moving up to the less easy ones. This results from these agents leveraging automatic curriculum learning, using their LP estimation as a proxy to choose goals that are at an affordable level of complexity. However, as opposed to semantic goals, there is less interference between classes and transfer is poorer (per-class SR increases sequentially). On the one hand, C-DS agents are unable to go beyond the $\widetilde{S}_2$ class. This further supports the idea that the node update step alone in deep sets does not provide enough representational power. On the other hand, both C-GN and C-IN manage to reach goals in all the evaluation classes, from no stacks at all to stacks of 5 objects. However, they are both unable to maximize their per-class performance. This suggests that learning policies that can achieve all evaluation classes at the same time with continuous goals is difficult and requires more training budget.

4.2 Ablation studies: How important is the ZPD management scheme ?



(a) Global SR. Stars highlight statistical differences w.r.t C-GN agents.

(b) Per class SR.

Figure 7: Performance metrics for C-GN, C-IN and their ZPD ablations. Mean $\pm$ standard deviations are computed over 5 seeds.

To assess the importance of the ZPD management mechanism for continuous goals, we consider the C-GN and C-IN agents — the best performing GNN-based architectures so far — and remove the biased initialization scheme: blocks are placed without any initial stacks in the resulting ablations. Figure 7 shows performance metrics for these agents. The global SR of both ablations increases slower than that of C-GN and C-IN (Figure 7b). Besides, it gets stuck at around 50% of the maximal performance while their corresponding full versions manage to reach 75%. Zooming on the per-class performance metrics shows the considerable decrease in the behavioral diversity when removing the ZPD management scheme (Figure 7b): the ablations struggle to transfer between easy goals ($\widetilde{S}_2$ and $\widetilde{S}_3$) and harder ones ($\widetilde{S}_4$ and $\widetilde{S}_5$).

5 Conclusion and Future Work

In this paper, we study several GNN-based goal-conditioned architectures for both the policy and critic in multi-object manipulation domains. More specifically, we considered four different computational schemes: full graph networks, interaction networks, relation networks and deep sets. We evaluated our agents using two different goal space structures: 1) continuous geometric goal spaces corresponding to per-object target positions; 2) semantic relational goal spaces based on the binary predicates close and above. Our study exhibits three main results. First, object-centered architectures induced with sufficiently strong representational capabilities are usually better suited than flat architectures which usually *struggle with entangled input features*. Second, the behavior of GNN-based architectures depends on the nature of the goal space. When the goal space already captures relational features between objects (semantic goals), interesting behaviors emerge even with schemes using weak representational capabilities. However, when goals are specific to each object independently (continuous goals), only architectures with more computations can increase the behavioral diversity. Third, combining efficient object-centered architectures and relational goals yields the best transfer between goals, as skills learned in easy goals can be better adapted to more complex ones. However, when goals are not relational, the transfer from primitive to more complex skills is weak and additional ingredients such as ZPD management are required.

This study makes a step towards open-ended autotelic agents capable of efficient transfer between abstract goals. However, the agents studied here only leverage their physical interactions with the environment. This does not account for the extraordinary human capacities to learn from social interactions (Vygotsky, 1978; Bruner, 1973; Tomasello, 2009). We believe adding social learning mechanisms as suggested in (Sigaud et al., 2021) is a promising line of research towards more capable open-ended agents.

ACKNOWLEDGMENTS

This work was performed using HPC resources from GENCI-IDRIS (Grant 2020-A0091011875). The authors would like to thank Hugo Caselles-Dupré and Mohamed Chetouani for insightful discussions.

REFERENCES

Joshua Achiam and Shankar Sastry. Surprise-based intrinsic motivation for deep reinforcement learning. *arXiv preprint arXiv:1703.01732*, 2017.

Ahmed Akakzia, Cédric Colas, Pierre-Yves Oudeyer, Mohamed Chetouani, and Olivier Sigaud. Grounding language to autonomously-acquired skills via goal generation. *arXiv preprint arXiv:2006.07185*, 2020.

Ahmed Akakzia, Cédric Colas, Pierre-Yves Oudeyer, Mohamed Chetouani, and Olivier Sigaud. Grounding language to autonomously-acquired skills via goal generation. In *ICLR 2021*, 2021.

Muhannad Alomari, Paul Duckworth, David C. Hogg, and Anthony G. Cohn. Natural language acquisition and grounding for embodied robotic systems. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

Peter Battaglia, Razvan Pascanu, Matthew Lai, Danilo Jimenez Rezende, et al. Interaction networks for learning about objects, relations and physics. *Advances in neural information processing systems*, 29, 2016.

Peter W. Battaglia, Jessica B. Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, et al. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*, 2018.

Marc Bellemare, Sriram Srinivasan, Georg Ostrovski, Tom Schaul, David Saxton, and Remi Munos. Unifying count-based exploration and intrinsic motivation. *Advances in neural information processing systems*, 29:1471–1479, 2016.

Jerome S. Bruner. Organization of early skilled action. *Child development*, pp. 1–11, 1973.

Yuri Burda, Harri Edwards, Deepak Pathak, Amos Storkey, Trevor Darrell, and Alexei A Efros. Large-scale study of curiosity-driven learning. *arXiv preprint arXiv:1808.04355*, 2018.

Cédric Colas, Pierre Fournier, Mohamed Chetouani, Olivier Sigaud, and Pierre-Yves Oudeyer. Curious: intrinsically motivated modular multi-goal reinforcement learning. In *International conference on machine learning*, pp. 1331–1340. PMLR, 2019.

Cédric Colas, Tristan Karch, Nicolas Lair, Jean-Michel Dussoux, Clément Moulin-Frier, Peter Ford Dominey, and Pierre-Yves Oudeyer. Language as a cognitive tool to imagine goals in curiosity-driven exploration. *arXiv preprint arXiv:2002.09253*, 2020.

Lisandro D. Dalcin, Rodrigo R. Paz, Pablo A. Kler, and Alejandro Cosimo. Parallel distributed computing using python. *Advances in Water Resources*, 34(9):1124–1139, 2011.

Jeffrey L Elman. Learning and development in neural networks: The importance of starting small. *Cognition*, 48(1):71–99, 1993.

Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural message passing for quantum chemistry. *arXiv preprint arXiv:1704.01212*, 2017.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *arXiv preprint arXiv:1801.01290*, 2018.

Johannes Kulick, Marc Toussaint, Tobias Lang, and Manuel Lopes. Active learning for teaching a robot grounded relational symbols. In *Twenty-Third International Joint Conference on Artificial Intelligence*, 2013.

John B Lanier, Stephen McAleer, and Pierre Baldi. Curiosity-driven multi-criteria hindsight experience replay. *arXiv preprint arXiv:1906.03710*, 2019.

Richard Li, Allan Jabri, Trevor Darrell, and Pulkit Agrawal. Towards practical multi-object manipulation using relational reinforcement learning. *arXiv preprint arXiv:1912.11032*, 2019.

Jean M Mandler. On the spatial foundations of the conceptual system and its enrichment. *Cognitive science*, 36(3):421–451, 2012.

Ashvin Nair, Vitchyr Pong, Murtaza Dalal, Shikhar Bahl, Steven Lin, and Sergey Levine. Visual reinforcement learning with imagined goals. *arXiv preprint arXiv:1807.04742*, 2018.

Deepak Pathak, Dhiraj Gandhi, and Abhinav Gupta. Self-supervised exploration via disagreement. In *International conference on machine learning*, pp. 5062–5071. PMLR, 2019.

Matthias Plappert, Marcin Andrychowicz, Alex Ray, Bob McGrew, Bowen Baker, Glenn Powell, Jonas Schneider, Josh Tobin, Maciek Chociej, Peter Welinder, et al. Multi-goal reinforcement learning: Challenging robotics environments and request for research. *arXiv preprint arXiv:1802.09464*, 2018.

Vitchyr H Pong, Murtaza Dalal, Steven Lin, Ashvin Nair, Shikhar Bahl, and Sergey Levine. Skew-fit: State-covering self-supervised reinforcement learning. *arXiv preprint arXiv:1903.03698*, 2019.

Rémy Portelas, Cédric Colas, Lilian Weng, Katja Hofmann, and Pierre-Yves Oudeyer. Automatic curriculum learning for deep RL: A short survey. *arXiv preprint arXiv:2003.04664*, 2020.

Adam Santoro, David Raposo, David G Barrett, Mateusz Malinowski, Razvan Pascanu, Peter Battaglia, and Timothy Lillicrap. A simple neural network module for relational reasoning. *Advances in neural information processing systems*, 30, 2017.

Franco Scarselli, Sweah Liang Yong, Marco Gori, Markus Hagenbuchner, Ah Chung Tsoi, and Marco Maggini. Graph neural networks for ranking web pages. In *The 2005 IEEE/WIC/ACM International Conference on Web Intelligence (WI'05)*, pp. 666–672. IEEE, 2005.

Olivier Sigaud, Hugo Caselles-Dupré, Cédric Colas, Ahmed Akakzia, Pierre-Yves Oudeyer, and Mohamed Chetouani. Towards teachable autonomous agents. *arXiv preprint arXiv:2105.11977*, 2021.

Luc Steels. The autotelic principle. In *Embodied artificial intelligence*, pp. 231–242. Springer, 2004.

Richard S Sutton, Andrew G Barto, et al. Reinforcement learning. *Journal of Cognitive Neuroscience*, 11(1):126–134, 1999.

Stefanie Tellex, Thomas Kollar, Steven Dickerson, Matthew R. Walter, Ashis Gopal Banerjee, Seth Teller, and Nicholas Roy. Approaching the symbol grounding problem with probabilistic graphical models. *AI magazine*, 32(4):64–76, 2011.

Michael Tomasello. *Constructing a language*. Harvard university press, 2009.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

Oriol Vinyals, Timo Ewalds, Sergey Bartunov, Petko Georgiev, Alexander Sasha Vezhnevets, Michelle Yeo, Alireza Makhzani, Heinrich Küttler, John Agapiou, Julian Schrittwieser, et al. Starcraft ii: A new challenge for reinforcement learning. *arXiv preprint arXiv:1708.04782*, 2017.

L. S. Vygotsky. Tool and Symbol in Child Development. In *Mind in Society*, chapter Tool and Symbol in Child Development, pp. 19–30. Harvard University Press, 1978. ISBN 0674576292. doi: 10.2307/j.ctvjf9vz4.6.

Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Russ R Salakhutdinov, and Alexander J Smola. Deep sets. In *Advances in neural information processing systems*, pp. 3391–3401, 2017.

Vinicius Zambaldi, David Raposo, Adam Santoro, Victor Bapst, Yujia Li, Igor Babuschkin, Karl Tuyls, David Reichert, Timothy Lillicrap, Edward Lockhart, et al. Relational deep reinforcement learning. *arXiv preprint arXiv:1806.01830*, 2018.

A Appendix

A.1 Pseudo code

Algorithms 1 and 2 present the high-level pseudo-code for the autotelic learning mechanism with respectively semantic and continuous goals.

Algorithm 1 Learning Semantic Goals

Require Env E,
Initialize policy Π, Uniform goal sampler $\mathcal{G}^s_{unif}$, buffer B.
discovered_goals = []
loop
    $g \leftarrow \mathcal{G}^s_{unif}$.sample_goal(discovered_goals)
    $trajectory \leftarrow E$.rollout(g)
    $\mathcal{G}^s_{unif}$.update($trajectory$)
    B.update($trajectory$)
    Π.update(B)
return Π

Algorithm 2 Learning Continuous Goals

Require Env E, Goal classes C_g
Initialize policy Π, LP-based goal sampler $\mathcal{G}^s_{LP}$, buffer B.
loop
    $c \leftarrow \mathcal{G}^s_{LP}$.sample_class($C_g$)
    $g \leftarrow generate\_positions(c)$
    $trajectory \leftarrow E$.rollout(g)
    $\mathcal{G}^s_{LP}$.update($trajectory$)
    B.update($trajectory$)
    Π.update(B)
return Π

A.2 Implementation details

In this part, we present details necessary to reproduce our results. We further open-source our code at https://github.com/akakzia/rlgraph.

GNN-*based networks.* Our four graph-based architectures use at most two shared networks, NN_{edge} and NN_{node}, respectively for computing updated edge features and node features. Both are 1-hidden-layer networks of hidden size 256. Taking the output dimension to be equal to $3\times$ the input dimension for the shared networks showed better results. All networks use ReLU activation and the Xavier initialization. For edge-wise and node-wise aggregation, we use a one-headed self-attention module. Finally, to produce the output, all architecture use a readout network NN_{out}. The latter is also a 1-hidden-layer network of hidden size 256. For optimization, we use Adam with learning rates 10^{-3}. The list of hyperparameters is provided in Table 1.

Parallel implementation of SAC-HER. All our experiments are based on a *Message Passing Interface* (Dalcin et al., 2011) to exploit multiple processors. Each of the 24 parallel workers maintains its own replay buffer of size 10^6 and performs its own updates. To synchronize experience between different workers, updates are summed over the 24 actors and the updated actor and critic networks are broadcast to all workers. Each worker alternates between 2 data collection episodes and 30 updates with batch size 256. To form an epoch, this cycle is repeated 50 times and followed by the offline evaluation of the agent.

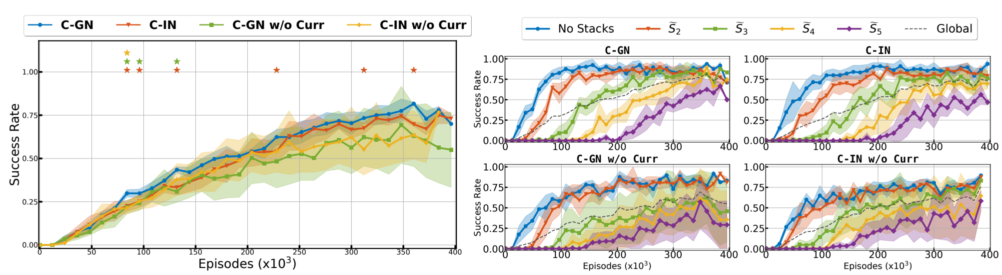
Table 1: Hyperparameters used in this paper.

Hyperparam.	Description	Values.
nb_mpis	Number of workers	24
nb_cycles	Number of repeated cycles per epoch	50
$nb\_rollouts\_per\_mpi$	Number of rollouts per worker	2
$rollouts\_length$	Number of episode steps per rollout	200
$nb\_updates$	Number of updates per cycle	30
$replay\_strategy$	HER replay strategy	$future$
k_replay	Ratio of HER data to data from normal experience	4
$batch\_size$	Size of the batch during updates	256
γ	Discount factor to model uncertainty about future decisions	0.99
τ	Polyak coefficient for target critics smoothing	0.95
lr_actor	Actor learning rate	10^{-3}
lr_critic	Critic learning rate	10^{-3}
α	Entropy coefficient used in SAC	0.2
$biased\_init$	Probability of following ZPD management scheme	0.2
$self\_eval\_curriculum$	Probability to perform self evaluations	0.1
$curriculum\_queue\_length$	Window over which LP estimations are made	1000

B Additional results

In this section, we present additional results which complement the ones presented in the main paper. More specifically, we study the relative importance of curriculum learning when using continuous goals (Appendix B.1) and of self-attention when using semantic goals (Appendix B.2).

B.1 Curriculum ablation



(a) Global SR. Stars highlight statistical differences w.r.t C-GN agents.

(b) Per class SR.

Figure 8: Performance metrics for C-GN, C-IN and their curriculum ablations. Mean $\pm$ standard deviations are computed over 5 seeds.

To study the relative importance of the LP-based curriculum learning mechanism used with continuous goals, we introduce ablations of C-GN and C-IN which uniformly sample a class of goals without any particular prioritization. We only consider architectures based on GN and IN in this ablation study since they show the best results in Section 4.1. Figure 8 presents the global performance metrics for C-GN, C-IN and their ablation counterparts. Autotelic agents using continuous goals but no curriculum clearly show an increased variance in their global performance. Figure 8 zooms on the local performance on each class for the considered agents. Compared to C-GN and C-IN, the shaded areas in the ablations are larger, suggesting that the learning process of the latter agents is not stable. Precisely, this is true in stacks of size 3 or higher. In fact, ablations face catastrophic forgetting as they engage with harder goals. The curriculum learning mechanism helps stabilize the learning process by focusing on goals of moderate level of complexity, including the ones that the agents are likely to forget during training. Note that this issue is specific to continuous goals, which shows that they are not well suited to transfer between different goals.

B.2 SELF-ATTENTION ABLATION

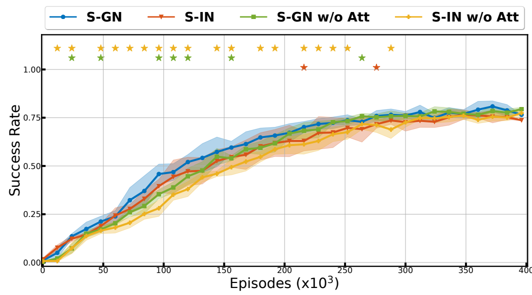


Figure 9: Global SR across training episodes for S-GN, S-IN and their self-attention ablations counterparts. Mean $\pm$ standard deviations are computed over 5 seeds. Stars highlight statistical differences w.r.t S-GN agents (Welch's t-test with null hypothesis $\mathcal{H}_0$: no difference in the means, $\alpha = 0.05$.

We propose to remove the self-attention aggregation schemes from S-GN and S-IN,– the two best performing agents,– and introduce the corresponding ablations which use an unweighted sum when performing the pooling over edges or nodes. Figure 9 presents the global SR for these agents across training episodes. The differences only appear at the beginning of training. In fact, the global performance metrics in the ablations increases slower than their corresponding full-versions (blue vs green; red vs orange). However, all agents seem to behave similarly by the end of training. This suggests that self-attention improves sample efficiency, yielding GNN-based agents that can faster capture actionable relational features within their graphs.