Learning to plan with uncertain topological maps

Edward Beeching[1[0000−0003−1165−3464]], Jilles Dibangoye[1[0000−0001−8826−4438]], Olivier Simonin[1[0000−0002−3070−7790]], and Christian Wolf[2[0000−0001−9766−3211]]

[1] INRIA Chroma team, CITI Lab. INSA Lyon, France.
https://team.inria.fr/chroma/en/
[2] Université de Lyon, INSA-Lyon, LIRIS, CNRS, France.
{firstname.lastname}@insa-lyon.fr

Project page https://edbeeching.github.io/papers/learning_to_plan

Abstract. We train an agent to navigate in 3D environments using a hierarchical strategy including a high-level graph based planner and a local policy. Our main contribution is a data driven learning based approach for planning under uncertainty in topological maps, requiring an estimate of shortest paths in valued graphs with a probabilistic structure. Whereas classical symbolic algorithms achieve optimal results on noise-less topologies, or optimal results in a probabilistic sense on graphs with probabilistic structure, we aim to show that machine learning can overcome missing information in the graph by taking into account rich high-dimensional node features, for instance visual information available at each location of the map. Compared to purely learned neural white box algorithms, we structure our neural model with an inductive bias for dynamic programming based shortest path algorithms, and we show that a particular parameterization of our neural model corresponds to the Bellman-Ford algorithm. By performing an empirical analysis of our method in simulated photo-realistic 3D environments, we demonstrate that the inclusion of visual features in the learned neural planner outperforms classical symbolic solutions for graph based planning.

Keywords: Visual navigation, topological maps, graph neural networks

1 Introduction

A critical part of intelligence is navigation, memory and planning. An animal that is able to store and recall pertinent information about their environment is likely to exceed the performance of an animal whose behavior is purely reactive. Many control and navigation problems in partially observed 3D environments involve long term dependencies and planning. It has been shown that humans and other animals navigate through the use of waypoints combined with a local locomotion policy [55,22]. In this work, we mimic this strategy by proposing a hierarchical planner, which performs high-level long term planning using an uncertain topological map (a valued graph including visual features) combined

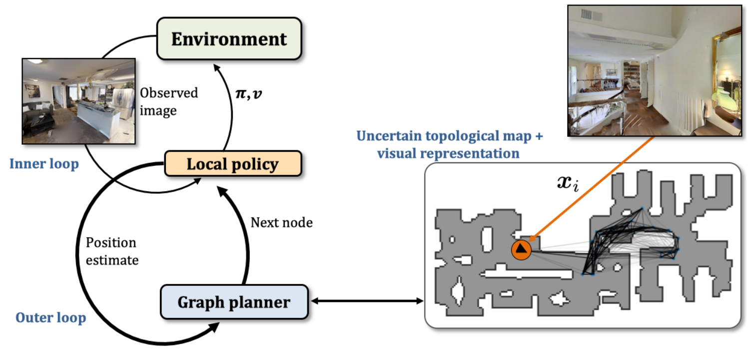

Fig. 1: An agent navigates to a goal location with a hierarchical planner. A high-level planner proposes target nodes in a topological map, which are used as an objective for a local point-goal policy. The graph is estimated from an explorative rollout and, as such, uncertain: the opacities of the edges correspond to estimations of connectivity between nodes (darker lines = higher confidence). We observe a low probability of connection between the node at the agent's position and its nearest neighbor, whereas from the visual observation associated to the node we can see there is a traversable space between the two nodes.

with a local RL-based policy navigating between high-level waypoints proposed by the graph planner. Our main contribution is a way to combine symbolic planning with machine learning, and we look to structure a neural network architecture to incorporate landmark based planning in unseen 3D environments.

When solving visual navigation tasks, biological or artificial agents require an internal representation of the environment if they want to solve more complex tasks than random exploration. We target a scenario where an agent is trained on a large-scale set of 3D environments to learn to reason on planning and navigation. When faced with a previously unseen environment, the agent is given the opportunity to build a representation by doing an explorative rollout from a previously learned explorative policy. It can then exploit this internal representation in subsequent visual navigation tasks. This corresponds to many realistic situations, where robots are deployed to indoor environments and are allowed to familiarize themselves before performing their tasks [43].

Our agent constructs an imperfect topological map of its environment, where nodes correspond to places and valued edges to connections. Edges are assigned two different values, spatial distances and probabilities indicating whether it is possible to navigate between the two nodes. Nodes are also assigned rich visual features extracted from images taken at the corresponding places in the environment. After deployment, the agent faces visual navigation tasks requiring it to find a specific location in the environment provided by a set of images corresponding to different viewpoints, extending the task proposed in [63]. The objective is to identify the goal location in the internal representation, and to provide an estimate for the shortest path to it. The main difficulty we address here is the fact that this path is an estimate only, since the ground truth path is not available during testing.

Whilst planning in graphs with known connectivity has been solved for many decades [15,7], planning under uncertainty remains an ongoing area of research. Whereas optimal results in a probabilistic sense exist for graphs with probabilistic connectivity, we aim to show that machine learning can overcome missing information in the graph by taking into account rich high-dimensional node features extracted from image observations associated with specific nodes. We train a graph neural network in a fully supervised way to predict estimates of the shortest path, using vision to overcome uncertainty in the connectivity information. We present a new variant of graph neural networks imbued with specific inductive bias, and we show that this structure can be parameterized to fallback to the classical Bellman-Ford algorithm.

Figure 1 illustrates the hierarchical planner: a neural graph based planner runs an outer loop providing estimates for next way-point on a graph, which are used as target nodes for a local RL-based policy running an inner loop and providing feedback to high-level planner on reached locations. Both planners take into account visual features, either stored in the graph (graph based planner), or directly as observations provided by the environment (local policy). The two planners are trained separately — the graph based planner in a fully supervised way from ground truth graphs, the local policy with RL and a point-goal strategy. This work makes the following contributions:

- A hierarchical model combining high-level graph based planning with a local point goal policy for robot navigation;
- A neural planner that combines an uncertain topological map with node features to learn to estimate shortest paths in noisy and unknown environments.
- A variant of graph networks encoding inductive bias inspired by dynamic programming-based shortest path algorithms.
- We evaluate this method in challenging and visually realistic 3D environments and show it outperforms symbolic planning on noisy topological maps.

2 Related work

Classical planning and graph search — A large body of work is available on classical planning on graphs, notable references include [31,42]. In robotics, there have been a number of works applying classical planning in topological maps for indoor robot navigation, for instance [48,54].

Planning under imperfect information — In many realistic robotic problems, the current state of the world is unknown. Though sensor observations provide measurements about the current state of the world, these measurements are usually incomplete or noisy because of disturbances that distort their values. Planning problems that face these issues are referred to as planning problems under imperfect information. Research on this topic has a long history, starting with

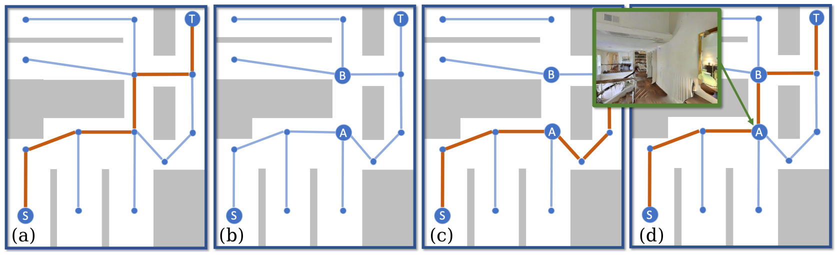

Fig. 2: Illustration of the different types of solutions to the high-level graph planning problem: (a) the ground truth graph (unavailable during testing) with the shortest path from node S to node T in red; (b) the uncertain graph available during test time. This graph is fully connected and for each edge a connection probability is available. For clarity we here show only edges where the connection probability is above a threshold. The edge from $A \rightarrow B$ is wrongly estimated as not connected; (c) an "optimal" path taking into account both probabilities and distances; (d) A learned shortest path, where the visual features at node A indicate passage to node B. We supervise a network to predict the GT path (a).

the seminal work by [2] presenting the first non-trivial exact dynamic programming algorithm for partially observable Markov decision processes (POMDPs). While there are other models [31, chap 12], POMDPs emerged as the standard framework to formalize and solve (single-agent) sequential decision-making problems with imperfect information about the state of the world [26]. As the agent does not have access to the actual state of the world, it acts based solely on its entire history of actions and observations, or the corresponding belief state, *i.e.,* the posterior probability distribution over the states given the history [2,50]. Approaches for finding optimal solutions have been investigated in the 2000s, ranging from dynamic programming [26] to heuristic search methods [51,30]. Key to these approaches is the idea that one can recast the original problem into a continuous-state fully observable Markov decision process, where states are belief states or histories [2]. Doing so allows theory and algorithm that applies for MDPs to also apply to POMDPs, albeit in much larger (and possibly continuous) state space. Another significant result of this literature is proof that the optimal value function is a piece-wise linear and convex function of the belief states, which allows the design of algorithms with faster rates of convergence [50]. For a thorough discussion on existing solvers for POMDPs, the reader can refer to [47].

Deep Reinforcement Learning — The field of Deep Reinforcement Learning (RL) has gained attention with successes on board games [49] and Atari games [37]. Recent works have applied Deep RL for the control of an agent in 3D environments [36] [24], exploring the use of auxiliary tasks such as depth prediction, loop detection and reward prediction to accelerate learning. Other recent work uses street-view scenes to train an agent to navigate in city environments [35]. To infer long term dependencies and store pertinent information about the partially observable environment, network architectures typically incorporate recurrent memory such as Gated Recurrent Units [13] or Long Short-Term Memory [23]. Extensions to memory based neural approaches began with Neural Turing Machines [19] and Differentiable Neural Computers [20], and have since been adapted to expand the capacity of Deep RL agents [56]. Spatially structured memory architectures have been shown to augment an agent's performance in 3D environments and are broadly split into two categories: metric maps which discretize the environment into a grid based structure and topological maps which produce node embeddings at key points in the environment. Research in learning to use a metric map is extensive and includes spatially structured memory [40], Neural SLAM based approaches [61] and approaches incorporating projective geometry and neural memory [21,8], these techniques are combined, extended and evaluated in [6]. Other works include that of Value Iteration Networks (VIN) [53] which approximate the value iteration algorithm with a CNN, applied planning in small fully observable state spaces (grid worlds). While VIN and our work structure planners, VINs use convolutions to approximate classical value iteration, while we use a graph representation and a novel GNN architecture with recurrent updates to approximate the Bellman-Ford algorithm. [27] plans under uncertainty in partially observable gridworld environments. Here uncertainty refers to POMPs, the classical QMDP algorithm is used as inductive bias for a neural network, whereas in our work uncertainty is over node connectivity in a graph constructed in a previously unseen environment. [52] is applied in observable state spaces to learn a forward model in a latent space to plan appropriate actions; they are not hierarchical, are not graph-based and do not appear to plan under uncertainty.

Research combining learning, navigation in 3D environments and topological representations has been limited in recent years with notable works being [43] who create graph a through random exploration in ViZDoom RL environment [28]. [16] performs planning in 3D environments on a graph-based structure created from randomly sampled observations, with node distances estimated with value estimates. The downside of these approaches is that to generalize to an unseen environment, many random samples must be taken to populate the graph.

Graph neural networks — Graph Neural Networks (GNN) are deep networks that operate on graphs directly. They have recently shown great promise in domains such as knowledge graphs [44], chemical analysis [18], protein interactions [17], physics simulations [3] and social network analysis [29]. These types of architectures enable learning from both node features and graph connectivity. Several review papers have covered graph neural networks in great detail [9,4,57,62]. GNNs have been applied to shortest path planning in travelling salesmen problems [33,25] and it has been reasoned that they can approximate optimal symbolic planning algorithms such as the Bellman-Ford algorithm [60]. This work applies a novel variant of GNN in order to solve approximate planning problems, where classical methods may struggle to deal with uncertainty.

3 Hierarchical navigation with uncertain graphs

We train an agent to navigate in a 3D visual environment and to exploit an internal representation, which it is allowed to obtain from an explorative rollout before the episode. Our objective is image goal, i.e. target-driven navigation to a location which is provided through a (visual) image. We extend the task introduced in [63] by generalizing to unseen environment configurations without the need to retrain the agent for a novel environment.

From the explorative rollout obtained with an agent trained with RL, which is further described in section 3.3, we create an uncertain topological map covering the environment, i.e. a valued graph $\mathcal{G}=\{\mathcal{V}, \boldsymbol{V}, \boldsymbol{E}, \boldsymbol{L}, \boldsymbol{D}\}$, where $\mathcal{V}=\{1, \ldots N\}$ is a set of nodes, $\boldsymbol{V}$ is a $K \times N$ matrix of rich visual node features of dimensions K, $\boldsymbol{E} \in [0,1]^{N \times N}$ is a set of edge probabilities where $\boldsymbol{E}_{i,j}$ is the probability of having an edge between nodes i and j, $\boldsymbol{L}$ is a matrix of node locations and $\boldsymbol{D}$ is a distance matrix, where $\boldsymbol{D}_{i,j}$ is a distance between nodes i and j. While $\boldsymbol{D}$ encodes a distance in a path planning sense, $\boldsymbol{E}$ encodes the probability of j being directly accessible from i with obstructions. The uncertainty encoded by this probability can be considered to be a combination of aleatory variability, i.e. uncertainty associated with natural randomness of the environment, as well as epistemic uncertainty, i.e. uncertainty associated with variability in computational models for estimating the graph, in our case the explorative policy trained with RL and taking into account visual observations.

Once the topological map is obtained, the objective of the agent at each episode is to navigate to a location given an image, which is provided as additional observation at each time step. The agent acts in 3D environments like Habitat[34] (see section 5), receiving images of the environment as observations and predicting actions from a discrete space (*forward*, *turn left 10* °, *turn right 10* °). We propose a hierarchical planner performing actions at two different levels:

- **A high-level graph based planner** that operates on longer time scale τ and iteratively proposes new point-goals nodes p_g^τ that are predicted, by a Graph Neural Network, to be on the shortest path from the agent to the estimated location of the target image.
- **A local policy** that has been trained to navigate to a local point-goal p_g^τ, which has been provided by the high-level policy. The local policy operates for a maximum of m time-steps, where m is a hyper-parameter, set to 10. The agent has been trained with an additional *STOP* action, so that it can learn to terminate the local policy in the case that it reaches p_g^τ in under m steps.

The two planners communicate through estimated locations, the graph planner indicating the next waypoint to the local policy as a location, and the local policy providing an estimate of its location back to the high-level planner. The planner updates its current node estimate as the nearest node and planning continues.

3.1 High-level planning with uncertain graphs

The objective of the high-level planner is to estimate the shortest path from the current position $S \in \mathcal{V}$ in the graph to a terminal node $T \in \mathcal{V}$, whose identity is estimated as the node whose visual features are closest to the target image in cosine distance. Planning takes into account the distances between nodes encoded in $\boldsymbol{D}$ as well as estimated edge connectivity encoded in $\boldsymbol{E}$. As an edge (i, j) may have a large connection probability $\boldsymbol{E}_{i,j}$ but still be obstructed in reality, the goal is to learn a trainable planner parameterized by parameters θ, which takes into account visual features $\boldsymbol{V}$ to overcome the uncertainty in the graph connectivity. To this end, we assume the ground truth connectivity $\boldsymbol{E}^*$ available during training only. Figure 2 illustrates the different types of solutions this problem admits: the optimal shortest path is only available on ground truth data (Figure 2a), the objective is to use the noisy uncertain graph (Figure 2b) and provide an estimate of the optimal solution taking into account visual features (Figure 2d). This is unlike the optimal solution in a probabilistic sense calculated from a symbolic algorithm (Figure 2c).

We propose a trainable planner, which consists of a novel graph neural network architecture with dedicated inductive bias for planning. Akin to graph networks [5], the node embeddings are updated with messages over the edges, which propagate information over the full graph. While it has been shown that graph networks can be trained to perform planning [59], we aim to closely mimic the structure of the Bellman-Ford algorithm and we embue the planner with additional inductive bias and a supervised objective to explicitly learn to calculate shortest paths from data. To this end, each node i of the graph is assigned an embedding $\boldsymbol{x_i} = [\boldsymbol{v_i}, \boldsymbol{e_i}, t_i, \boldsymbol{d_i}, \boldsymbol{s_i}]$ where $\boldsymbol{v_i}$ are visual features from the memory matrix $\boldsymbol{V}$, t_i is a boolean value indicating if the node is the target, $\boldsymbol{e_i}$ are the edge connection probabilities from node i to all other nodes, $\boldsymbol{d_i}$ are the distances for node i to all other nodes, $\boldsymbol{s_i}$ is a one hot vector identifying the node.

We motivate our neural model with the following objective: the planner should be able to exploit information contained in the graph connectivity, but also in the visual features, to be able to find the shortest path from a given current node to a target node. As with classical planning algorithms, it will thus be required to keep for each node a latent representation of the bound d_i on the shortest distance as well as information on the identity of the outgoing edge to the neighbor on the shortest path, the predecessor function $\Pi(i)$. Known algorithms (Dijstra, Bellman-Ford) perform iterative updates of these variables (d_i, Π_i) by comparing them with neighboring nodes and their inter-node distances, updating the bound d_i and Π_i when a shorter path is found than the current one. This is usually done by iterating over the successors of a given node i.

In our trained model, these variables are not made explicit, but they are supposed to be learned as a unique vectorial latent representation for each node i in the form of an internal state $\boldsymbol{r_i}$, which generally holds current information on the reasoning of the agent. The input to each iteration of the graph network is, for each node i, the node embedding $\boldsymbol{x_i}$, and the node state $\boldsymbol{r}_i$, which we concatenate to form a single node vector $\boldsymbol{n_i} = [\boldsymbol{x}_i, \boldsymbol{r}_i] = [\boldsymbol{v_i}, \boldsymbol{e_i}, t_i, \boldsymbol{d_i}, \boldsymbol{s_i}, \boldsymbol{r_i}]$. As classically done in graph neural networks, this representation is updated iteratively by exchanging messages between nodes in the form of trainable functions. The messages and trainable functions of our model are given as follows, illustrated in

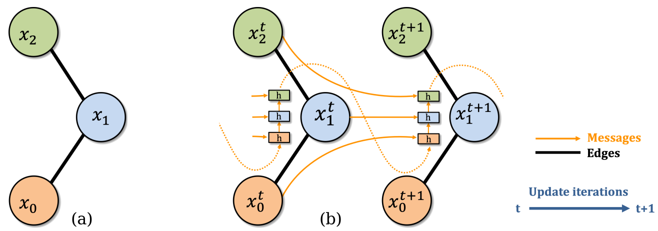

Fig. 3: (a) An example graph; (b) One iteration of the neural planner's message passing and bound update. Messages from neighbors are serialized and fed through a GRU, an inductive bias for learning minima necessary for bound updates.

Figure 3, and will be motivated in detail further below.

$$\boldsymbol{m}_{i,j} = \boldsymbol{W}_1[\boldsymbol{n}_i, \boldsymbol{n}_j] \odot \sigma(\boldsymbol{W_2}[\boldsymbol{n}_i, \boldsymbol{n}_j]) \tag{1}$$

$$\boldsymbol{r}_i' = \phi^{r \leftarrow h}(\{\boldsymbol{m}_{i,j}\}_{\forall j}, \boldsymbol{h}_i) \tag{2}$$

Here, $\odot$ is the Hadamard product, $\boldsymbol{W}_.$ are weight matrices, and $\boldsymbol{r}_i'$ is the updated latent representation. The features $\boldsymbol{x}_i$ do not change during these operations.

Equation (1) is inspired from gated linear layers [14], and enables each node to identify whether it is the target, and update its representation of the bound. We use gated linear layers in order to provide the network with the capacity to update bound estimates for its neighbors.

Equation (2) integrates messages from all neighbors j of node i, updating its latent representation. Since planning requires this step to update internal bounds on shortest paths, akin to shortest path algorithms that rely on dynamic programming, we serialize the updates from different neighbors into a sequence of updates, which allows the network to learn to calculate minimum functions on bound estimates. In particular, we model this through a Gated Recurrent Unit [12], using a hidden state vector $\boldsymbol{h}_i$ associated to each node i. The step is structured to mimic the min operation of the Bellman-Ford algorithm (see section 3.2 for details on this equivalence).

Equation (2) can thus be rewritten in more detail as follows: Going sequentially over the different neighbors j of node i, the hidden state $\boldsymbol{h}_i$ is updated as follows:

$$\boldsymbol{h}_i^{[j]} = \boldsymbol{W}_3 \boldsymbol{m}_{i,j} + \boldsymbol{W}_4 \boldsymbol{h}_i^{[j-1]} \tag{3}$$

For simplicity, we omitted the gating equations of GRUs and presented a single layer GRU. In practice we include all gating operations and use a stacked GRU with two layers. The output of the recurrent unit is a non-linear function of the last hidden state, providing the new latent value $\boldsymbol{r}_i' = MLP(\boldsymbol{h}_i^N)$.

The above messages are exchanged and accumulated for k steps where k is a hyper-parameter which should be at least the largest span of the graphs in the dataset. The action distribution is then estimated for each node as a linear mapping of the node embeddings followed by a softmax activation function.

3.2 Relations to optimal symbolic planners

As mentioned before, our neural planner could in theory be instantiated with a specific set of network parameters such that it corresponds to a known symbolic planner calculating an optimal path in a certain sense. To illustrate the relationship of the network structure, in particular the recurrent nature of the graph updates, we will layout details for the case where the planner performs the estimation of a shortest path given the distance matrix and ignoring the uncertainty information — an adaptation to an optimal planner in the probabilistic sense can be done in a straightforward manner. To avoid misunderstandings, we insist that the reasoning developed in this sub section is for illustration and general understanding of the chosen inductive network bias only, the real network parameters are fully trained with supervised learning as explained in section 4.

Handcrafting a parameterization requires imposing a structure on the node state $\boldsymbol{r}_i$, which otherwise is a learned representation. In our case, the node state will be composed of the bound b_i on the shortest path from the given node to the target node (a scalar), and the current estimate Π_i of the identity of predecessor node of node i w.r.t. the shortest path, which can be represented as a 1-in-K encoded vector indicating a distribution over nodes. Standard Bellman-Ford iteratively updates the bound for a given node i by examining all its neighbors j and checking whether a shorter path can be found passing through neighbor j. This can be written in a sequential form s.t. the bound gets updated iterating through the neighbors j=1...J_i of node i:

$$\begin{aligned} b_i^{[0]} &= b_i \\ b_i^{[j]} &= \min(b_i^{[j-1]}, b_j + d_{ij}) \\ b_i' &= b_i^{[J_i]} \end{aligned} \tag{4}$$

where b_i and b_i' are the bounds before and after the round of updates for node i.

In our neural formulation, the message updates given in equation (2), further developed in (3), mimic the Bellman-Ford bound update given in Equations (4). This provided motivation for our choice of a recurrent neural network in the graph neural network, as we require the update of the recurrent state h_i^j in Equation (3) to be able to perform a minimum operation and an arg min operation (or differentiable approximations of min and arg min).

3.3 Graph creation from explorative rollouts

Graphs were generated during the initial rollout from an exploratory policy trained with Reinforcement Learning. During training, the agent interacts with training environments and receives RGB-D image observations calculated as a projection from the 3D environment. The agent is trained to explore the environment and to maximize coverage, similar to [10,11].

To learn to estimate the graph connectivity, we add an auxiliary loss to the agent's objective function, $f_{link}(\boldsymbol{o_i}, \boldsymbol{o_j}, \boldsymbol{h_i})$ which is trained to classify whether two locations are in line of sight of each other, conditioned on the visual features

$\boldsymbol{o_i}, \boldsymbol{o_j}$ from the two locations and the agent's hidden state $\boldsymbol{h_i}$. Node features were calculated with a CNN [32]. Ground truth line of sight measurements were computed by 2D ray tracing on an occupancy map of each environment. In order to limit the size of the graph to a maximum number of nodes k, we aim to maximize each node's coverage of the environment using a Gaussian kernel function. At each time step a new node is observed by the agent, previous node positions are compared with a Gaussian kernel function (eq. 5) in order to identify the index of the most redundant node r, which is removed from the graph and replaced with the new node, node connectivities are then recomputed with $f_{link}(.)$, where $\boldsymbol{L}_i$ is the location of node i.

$$r = \arg \min_i \sum_j K(\boldsymbol{L}_i, \boldsymbol{L}_j), \qquad K(\boldsymbol{v}, \boldsymbol{v'}) = \exp(\frac{-\|\boldsymbol{v} - \boldsymbol{v'}\|^2}{2\sigma^2}), \tag{5}$$

4 Training

The high-level graph based planner — is trained in a purely supervised way. We generate ground truth labels by running a symbolic algorithm (Dijkstra [15]) on a set of valued ground truth training graphs described with the method detailed in Section 3.3. In particular, the supervised training algorithm takes as input *uncertain/noisy* graphs, which include visual features, and is supervised to learn to produce paths, which are calculated from known ground truth graphs unavailable during test time. During training we treat path planning as a classification problem where for a given target, each node must learn to predict the subsequent node on the optimal path to the target. Formally for each node i we predict a distribution $\boldsymbol{A_i}$ and aim to match a ground-truth distribution $\boldsymbol{A_i^*}$, which is a one-hot vector, minimizing cross entropy loss $\mathcal{L}(\boldsymbol{A}, \boldsymbol{A^*}) = -\sum_{i=1}^{n} \boldsymbol{A}_i^* \log \boldsymbol{A}_i$.

We augment training with a novel version of mod-drop [39], an algorithm for multi-modal data, which drops modalities probabilistically during training. During training we extend the node connection probabilities with the ground truth node adjacencies and mask either the probabilities or the adjacencies with a probability of 50%, during training we linearly taper the masking probability from 50% to 100% over the first 250 epochs. Ensuring that the final model requires only connection probabilities, but the reasoning performed during message passing and recurrent updates can be bootstrapped from the ground truth adjacencies. Training curves on unseen validation data are shown in figure 5.

The local policy — is a recurrent version of AtariNet [38] with two output heads for the action distribution and value estimates. The network was trained with a reinforcement learning algorithm Proximal Policy Optimization (PPO) [45] to navigate with discrete actions to a local point-goal. Point-goals were generated to be within 5m of the spawn location of agent. A dense reward was provided that corresponds to a decrease in geodesic distance to the target, a large reward (10.0) was provided when the agent reached the target and the *STOP* action was used. The episode was terminated when either the *STOP* action was used or after 500 time-steps. A negative reward of -0.01 was given at each time-step .

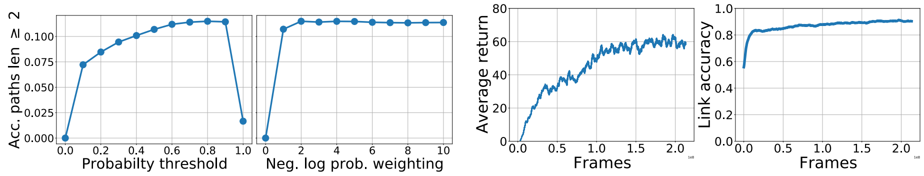

Fig. 4: Left: Symbolic baselines Dijkstra on thresholded probs. & cost function(5.1). Right: Average return & Acc. of line of sight predictions.

Table 1: H-SPL and acc. of the neural planner's predictions.

(a) Uncertain graphs

Method	Acc	H-SPL
Symbolic (threshold)	0.114	0.184
Symbolic (custom cost)	0.115	0.269
Neural (w/o visual)	0.251	0.468
Neural (w visual)	**0.262**	**0.501**

(b) Ground truth graphs

Method	Acc	H-SPL
Symbolic (GT)	**1.00**	**1.00**
Neural planner (GT)	0.921	0.983

The explorative policy for graph creation — is trained with PPO [46]. We aim to maximize coverage that is within the field of view of the agent. We create an occupancy grid of the environment with a grid spacing of 10cm. The first time a cell is observed the agent receives a reward of 0.1. A cell is considered to observable if it is free space, within 3m of the agent and in the field of view of the agent. Agent performance is shown in Figure 4.

5 Experiments

We evaluated our method in simulated 3D environments in the Habitat [34] simulator with the visually realistic Gibson dataset [58]. During training, the agent interacts with 72 different environments, where each environment corresponds to a different home. We evaluate on a set of 16 held out environments.

5.1 High-level graph-based planner

The neural planner was implemented in PyTorch [41].We compare two metrics, accuracy of prediction of the next way-point along the optimal path and the SPL metric [1], both for paths of length two or greater. As we evaluate SPL for both the high level planner and the hierarchical planner-controller, we refer to the high-level planner's SPL as H-SPL to avoid ambiguity.

Symbolic baselines — We compare the neural planner to two symbolic baselines, which reason on the uncertain graph only, without taking into account rich node features. While these baselines are *"optimal"* with respect to their respective objective functions, they are optimal with respect to the amount of information available to them, which is uncertain: *(i) Thresholding* — In order

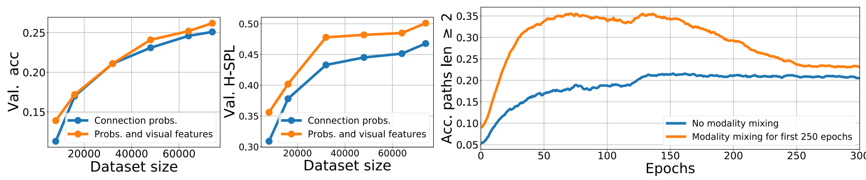

Fig. 5: Left & centre: Accuracy and H-SPL with increasing size of data when training with and without visual features. Right: Modality mixing.

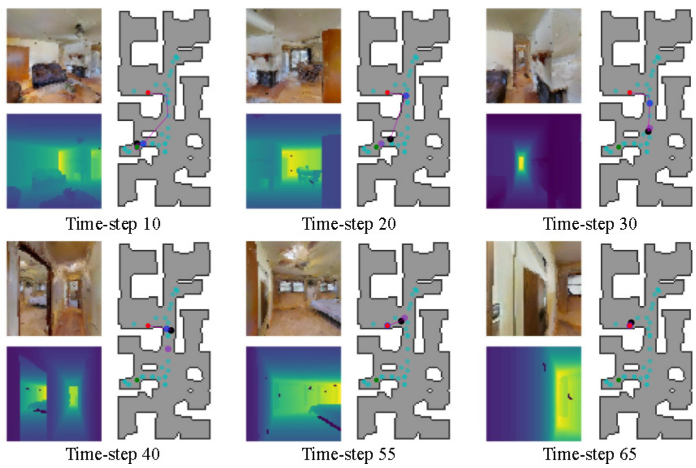

Fig. 6: Time-steps from a rollout of the hierarchical planner (graph+local). For each time-step: left – RGB-D obs., right – map of the environment (unseen) with graph nodes, source node, target node, agent position(black), agent's nearest neighbour, local point-goal provided by the planner and planned path.

to generate non-probabilistic edge connections, we threshold the connection probabilities with values ranging from 0-1 in steps of 0.1. After threshholding the graph, path planning was performed with Dijkstra's algorithm; *(ii) A custom cost function* for Dijkstra's algorithm weighting distances and probabilities: $cost(i,j) = \boldsymbol{D}_{i,j} - \lambda \log(\boldsymbol{E}_{i,j})$. We vary λ in order to control the trade-off of distance and connection probability. When λ is 0, the graph is a fully connected graph, whereas high values of λ would lead to finding the most probable path.

Results of both symbolic baselines are shown in Figure 4, we observe that they perform poorly under uncertainty. We evaluate the accuracy of their predictions with respect to the symbolic baseline on the ground truth graph. We report accuracy on source-target pairs separated by at least 2 steps. **Image driven recurrent baseline** — We also compare to an end-to-end RL approach where the current obs. and target are provided to a CNN based RL agent. The

Table 2: Performance of the hierarchical graph planner & local policy

Method: Planner + Local policy	**Success rate**	**SPL**
Graph oracle (optimal point-goals, not comparable)	0.963	0.882
Random	0.152	0.111
Recurrent Image-goal agent	0.548	0.248
Symbolic (threshold)	0.621	0.527
Symbolic (custom cost)	0.707	0.585
Neural planner (sampling)	0.966	0.796
Neural planner (deterministic)	**0.983**	**0.877**

architecture is a siamese CNN with a recurrent GRU. We train with a dense reward of improvement in geodesic dist. and provide a reward of 10 when the agent reaches the goal. We train with PPO for 200 M frames.

In Table 1a, we compare the neural planner with the baselines. We can see, that even without visual features, the neural planner is able to outperform the "optimal" symbolic baselines. This can be explained with the fact, that the baselines optimize a fixed criterion, whereas the neural planner can learn to exploit patterns in the connection probability matrix $\boldsymbol{E}$ to infer valuable information on shortest ground truth path. The gap further increases when the neural planner can use visual features. Results of modality mixing (see section 4) are shown in Figure 5. As a sanity check, table 1b compares the optimal symbolic planner against the neural planner trained with ground truth adjacencies provided as input. The results of the neural planner are close to optimum in this case.

We evaluated our approach on dataset sizes ranging from 8,000 graphs to 74,000 graphs (Figure 5). One graph contains 32×32 possible source-target combinations, leading to a maximum amount of 75,000,000 training instances.

5.2 Hierarchical planning and control (topological & local policy)

We evaluated the neural graph planner coupled with the local policy. For a given episode, the graph planner estimates the next node in the path to a target image and provides its location to the local policy, which executes for m time-steps. The planner then re-plans from the nearest neighbor to the agent's current position, this back and forth process of planning and navigating continues until either the agent reaches the target or 500 low-level time-steps have been conducted. We report accuracy as percentage of runs completed successfully and SPL in table 2, albeit measured on low-level trajectories as opposed to graph space. We combine the local policy with various graph planners, and can see that the neural graph planners greatly outperform the symbolic baselines. We perform two evaluations of the neural planner; a deterministic evaluation where point-goals are chosen with the argmax of the $\boldsymbol{A}$ distribution and a non-deterministic one by sampling from $\boldsymbol{A}$. The motivation is that by sampling, the planner can escape from local minima and loops created by errors in approximation. This is confirmed when studying rollouts from the agents, and also quantitatively through the performances shown

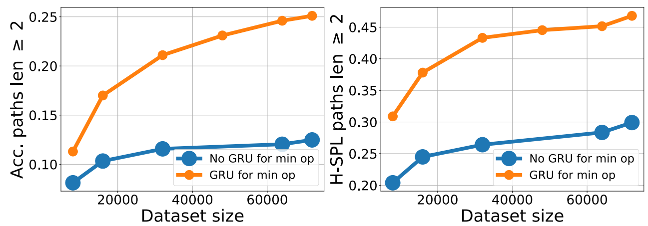

Fig. 7: Ablation of a GRU for the accumulation of incoming messages. The GRU was added to ensure that the model could represent the Bellman-Ford algorithm.

in table 2. A visualization of steps from an episode is shown in Figure 6, where in step 10 we can see that navigation is robust w.r.t. local errors in planning.

5.3 Ablation: Effect of chosen inductive bias

As developed in sections 3.1 and 3.2, our graph based planner includes a particular inductive bias, which allows it to represent the Bellman-Ford algorithm for the calculation of shortest or best paths. This bias is implemented as a recurrent model (a GRU) running sequentially over the message passing procedures, as illustrated in Figure 3. Figure 7 ablates the effect of this additional bias as a function of data sizes ranging from 8,000 to 74,000 example graphs, each one evaluated with 32^2=1,024 different combinations of starting and end points.

The differences are substantial, we can see that our model is able to exploit increasing amounts of data and translate them into gains in performance, whereas standard graph convolutional networks do not — we conjecture that they lack in structure allowing them to pick up the required reasoning.

6 Conclusion

We demonstrated that path planning can be approximated with learning when structured in a manner that is akin to classical path planning algorithms. We have performed an empirical analysis of the proposed solution in photo-realistic 3D environments and have shown that in uncertain environments graph neural networks can outperform their symbolic counterparts by incorporating rich visual features. Our method can be used to augment a vision based agent with the ability to form long term plans under uncertainty in novel environments, without a priori knowledge of the particular environment. We have analysed the empirical performance of the neural planning algorithm with a variety of dataset sizes, shown that the high-level planner can be coupled with a low-level policy and evaluated the hierarchical performance on an image-goal task.

Acknowledgements — This work was funded by grant Deepvision (ANR-15-CE23-0029, STPGP479356-15), a joint French/Canadian call by ANR & NSERC; Compute was provided by the CNRS/IN2P3 Computing Center (Lyon, France), and by GENCI-IDRIS (Grant 2019-100964).

References

1. Anderson, P., Chang, A., Chaplot, D.S., Dosovitskiy, A., Gupta, S., Koltun, V., Kosecka, J., Malik, J., Mottaghi, R., Savva, M., Zamir, A.R.: On evaluation of embodied navigation agents (2018)
2. Åström, K.J.: Optimal control of markov processes with incomplete state information. Journal of Mathematical Analysis and Applications **10**(1), 174–205 (1965)
3. Battaglia, P., Pascanu, R., Lai, M., Rezende, D.J., et al.: Interaction networks for learning about objects, relations and physics. In: Advances in neural information processing systems. pp. 4502–4510 (2016)
4. Battaglia, P.W., Hamrick, J.B., Bapst, V., Sanchez-Gonzalez, A., Zambaldi, V., Malinowski, M., Tacchetti, A., Raposo, D., Santoro, A., Faulkner, R., et al.: Relational inductive biases, deep learning, and graph networks. arXiv preprint arXiv:1806.01261 (2018)
5. Battaglia, P., Hamrick, J., Bapst, V., Sanchez-Gonzalez, A., Zambaldi, V., Malinowski, M., Tacchetti, A., Raposo, D., Santoro, A., Faulkner, R., Gülçehre, Ç., Song, F., Ballard, A., Gilmer, J., Dahl, G., Vaswani, A., Allen, K., Nash, C., Langston, V., Dyer, C., Heess, N., Wierstra, D., Kohli, P., Botvinick, M., Vinyals, O., Li, Y., Pascanu, R.: Relational inductive biases, deep learning, and graph networks. arXiv preprint 1807.09244 (2018)
6. Beeching, E., Wolf, C., Dibangoye, J., Simonin, O.: Egomap: Projective mapping and structured egocentric memory for deep rl (2020)
7. Bellman, R.: On a routing problem. Quarterly of applied mathematics **16**(1), 87–90 (1958)
8. Bhatti, S., Desmaison, A., Miksik, O., Nardelli, N., Siddharth, N., Torr, P.H.S.: Playing doom with slam-augmented deep reinforcement learning. arxiv preprint 1612.00380 (2016)
9. Bronstein, M.M., Bruna, J., LeCun, Y., Szlam, A., Vandergheynst, P.: Geometric deep learning: going beyond euclidean data. IEEE Signal Processing Magazine **34**(4), 18–42 (2017)
10. Chaplot, D.S., Gandhi, D., Gupta, S., Gupta, A., Salakhutdinov, R.: Learning to explore using active neural slam. In: International Conference on Learning Representations (2020), `https://openreview.net/forum?id=HklXn1BKDH`
11. Chen, T., Gupta, S., Gupta, A.: Learning exploration policies for navigation. In: International Conference on Learning Representations (2019), `https://openreview.net/forum?id=SyMWn05F7`
12. Chung, J., Gulcehre, C., Cho, K., Bengio, Y.: Empirical evaluation of gated recurrent neural networks on sequence modeling. arXiv preprint arXiv:1412.3555 (2014)
13. Chung, J., Gulcehre, C., Cho, K., Bengio, Y.: Gated Feedback Recurrent Neural Networks. In: ICML (2015)
14. Dauphin, Y.N., Fan, A., Auli, M., Grangier, D.: Language modeling with gated convolutional networks. In: Proceedings of the 34th International Conference on Machine Learning-Volume 70. pp. 933–941. JMLR. org (2017)
15. Dijkstra, E.W.: A note on two problems in connexion with graphs. Numerische mathematik **1**(1), 269–271 (1959)
16. Eysenbach, B., Salakhutdinov, R.R., Levine, S.: Search on the replay buffer: Bridging planning and reinforcement learning. In: Advances in Neural Information Processing Systems 32, pp. 15220–15231. Curran Associates, Inc. (2019)
17. Fout, A., Byrd, J., Shariat, B., Ben-Hur, A.: Protein interface prediction using graph convolutional networks. In: Advances in neural information processing systems. pp. 6530–6539 (2017)

18. Gilmer, J., Schoenholz, S.S., Riley, P.F., Vinyals, O., Dahl, G.E.: Neural message passing for quantum chemistry. In: Proceedings of the 34th International Conference on Machine Learning-Volume 70. pp. 1263–1272. JMLR. org (2017)
19. Graves, A., Wayne, G., Danihelka, I.: Neural turing machines. arXiv preprint arXiv:1410.5401 (2014)
20. Graves, A., Wayne, G., Reynolds, M., Harley, T., Danihelka, I., Grabska-Barwińska, A., Colmenarejo, S.G., Grefenstette, E., Ramalho, T., Agapiou, J., et al.: Hybrid computing using a neural network with dynamic external memory. Nature **538**(7626), 471 (2016)
21. Gupta, S., Davidson, J., Levine, S., Sukthankar, R., Malik, J.: Cognitive mapping and planning for visual navigation. In: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 7272–7281 (July 2017). https://doi.org/10.1109/CVPR.2017.769
22. Gupta, S., Fouhey, D., Levine, S., Malik, J.: Unifying map and landmark based representations for visual navigation. arXiv preprint arXiv:1712.08125 (2017)
23. Hochreiter, S., Schmidhuber, J.: Long Short-Term Memory. Neural Computation **9**(8), 1735–1780 (1997)
24. Jaderberg, M., Mnih, V., Czarnecki, W.M., Schaul, T., Leibo, J.Z., Silver, D., Kavukcuoglu, K.: Reinforcement learning with unsupervised auxiliary tasks. In: ICLR (2017)
25. Joshi, C.K., Laurent, T., Bresson, X.: An efficient graph convolutional network technique for the travelling salesman problem. arXiv preprint arXiv:1906.01227 (2019)
26. Kaelbling, L.P., Littman, M.L., Cassandra, A.R.: Planning and acting in partially observable stochastic domains. Artificial intelligence **101**(1-2), 99–134 (1998)
27. Karkus, P., Hsu, D., Lee, W.S.: Qmdp-net: Deep learning for planning under partial observability (2017)
28. Kempka, M., Wydmuch, M., Runc, G., Toczek, J., Jaskowski, W.: ViZDoom: A Doom-based AI research platform for visual reinforcement learning. IEEE Conference on Computatonal Intelligence and Games, CIG (2017). https://doi.org/10.1109/CIG.2016.7860433, `https://arxiv.org/pdf/1605.02097.pdf`
29. Kipf, T.N., Welling, M.: Semi-supervised classification with graph convolutional networks. In: International Conference on Learning Representations (2017)
30. KURNIAWATI, H.: Sarsop: Efficient point-based pomdp planning by approximating optimally reachable belief spaces. Proc. Robotics: Science and Systems, 2008 (2008)
31. LaValle, S.M.: Planning algorithms. Cambridge university press (2006)
32. Lecun, Y., Eon Bottou, L., Bengio, Y., Haaner, P.: Gradient-Based Learning Applied to Document Recognition. Proceedings of the IEEE **86**(11), 2278 – 2324 (1998)
33. Li, Z., Chen, Q., Koltun, V.: Combinatorial optimization with graph convolutional networks and guided tree search. In: Advances in Neural Information Processing Systems. pp. 539–548 (2018)
34. Manolis Savva*, Abhishek Kadian*, Oleksandr Maksymets*, Zhao, Y., Wijmans, E., Jain, B., Straub, J., Liu, J., Koltun, V., Malik, J., Parikh, D., Batra, D.: Habitat: A Platform for Embodied AI Research. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) (2019)
35. Mirowski, P., Grimes, M.K., Malinowski, M., Hermann, K.M., Anderson, K., Teplyashin, D., Simonyan, K., Kavukcuoglu, K., Zisserman, A., Hadsell, R.: Learning to Navigate in Cities Without a Map. arxiv pre-print 1804.00168v2 (2018)

36. Mirowski, P., Pascanu, R., Viola, F., Soyer, H., Ballard, A.J., Banino, A., Denil, M., Goroshin, R., Sifre, L., Kavukcuoglu, K., Kumaran, D., Hadsell, R.: Learning to Navigate in Complex Environments. In: ICLR (2017)
37. Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., Graves, A., Riedmiller, M., Fidjeland, A.K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., Hassabis, D.: Human-level control through deep reinforcement learning. Nature **518**(7540) (2015)
38. Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., Graves, A., Riedmiller, M., Fidjeland, A.K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., Hassabis, D.: Human-level control through deep reinforcement learning. Nature **518** (2015). https://doi.org/10.1038/nature14236, `https://storage.googleapis.com/deepmind-media/dqn/DQNNaturePaper.pdf`
39. Neverova, N., Wolf, C., Taylor, G., Nebout, F.: Moddrop: adaptive multi-modal gesture recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence **38**(8), 1692–1706 (2015)
40. Parisotto, E., Salakhutdinov, R.: Neural map: Structured memory for deep reinforcement learning. In: ICLR (2018)
41. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., Chintala, S.: Pytorch: An imperative style, high-performance deep learning library. In: Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché Buc, F., Fox, E., Garnett, R. (eds.) Advances in Neural Information Processing Systems 32, pp. 8024–8035. Curran Associates, Inc. (2019)
42. Remolina, E., Kuipers, B.: Towards a general theory of topological maps. Artif. Intell. **152**, 47–104 (2004)
43. Savinov, N., Dosovitskiy, A., Koltun, V.: Semi-parametric topological memory for navigation. In: International Conference on Learning Representations (2018), `https://openreview.net/forum?id=SygwwGbRW`
44. Schlichtkrull, M., Kipf, T.N., Bloem, P., Van Den Berg, R., Titov, I., Welling, M.: Modeling relational data with graph convolutional networks. In: European Semantic Web Conference. pp. 593–607. Springer (2018)
45. Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O.: Proximal policy optimization algorithms. arxiv pre-print 1707.06347 (2017)
46. Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O.: Proximal policy optimization algorithms. arXiv preprint arXiv:1707.06347 (2017)
47. Shani, G., Pineau, J., Kaplow, R.: A survey of point-based pomdp solvers. Autonomous Agents and Multi-Agent Systems **27**(1), 1–51 (2013)
48. Shatkay, H., Kaelbling, L.P.: Learning topological maps with weak local odometric information. In: IJCAI (2). pp. 920–929 (1997)
49. Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., Lanctot, M., Sifre, L., Kumaran, D., Graepel, T., Lillicrap, T., Simonyan, K., Hassabis, D.: A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. Science **362**(6419), 1140–1144 (2018)
50. Smallwood, R.D., Sondik, E.J.: The optimal control of partially observable markov processes over a finite horizon. Operations research **21**(5), 1071–1088 (1973)
51. Smith, T., Simmons, R.: Heuristic search value iteration for pomdps. In: Proceedings of the 20th conference on Uncertainty in artificial intelligence. pp. 520–527 (2004)

52. Srinivas, A., Jabri, A., Abbeel, P., Levine, S., Finn, C.: Universal planning networks (2018)
53. Tamar, A., Wu, Y., Thomas, G., Levine, S., Abbeel, P.: Value iteration networks (2016)
54. Thrun, S.: Learning metric-topological maps for indoor mobile robot navigation. Artificial Intelligence **99**(1), 21–71 (1998)
55. Wang, R.F., Spelke, E.S.: Human spatial representation: Insights from animals. Trends in Cognitive Sciences **6**(9), 376–382 (2002). https://doi.org/10.1016/s1364-6613(02)01961-7
56. Wayne, G., Hung, C.C., Amos, D., Mirza, M., Ahuja, A., Grabska-Barwinska, A., Rae, J.W., Mirowski, P.W., Leibo, J.Z., Santoro, A., Gemici, M., Reynolds, M., Harley, T., Abramson, J., Mohamed, S., Rezende, D.J., Saxton, D., Cain, A., Hillier, C., Silver, D., Kavukcuoglu, K., Botvinick, M., Hassabis, D., Lillicrap, T.P.: Unsupervised predictive memory in a goal-directed agent. arxiv preprint 1803.10760 (2018)
57. Wu, Z., Pan, S., Chen, F., Long, G., Zhang, C., Yu, P.S.: A comprehensive survey on graph neural networks. arXiv preprint arXiv:1901.00596 (2019)
58. Xia, F., R. Zamir, A., He, Z.Y., Sax, A., Malik, J., Savarese, S.: Gibson env: real-world perception for embodied agents. In: Computer Vision and Pattern Recognition (CVPR), 2018 IEEE Conference on. IEEE (2018)
59. Xu, K., Li, J., Zhang, M., Du, S., Kawarabayashi, K., Jegelka, S.: What can neural networks reason about? arxiv preprint 1905.13211 (2019)
60. Xu, K., Li, J., Zhang, M., Du, S.S., Kawarabayashi, K.i., Jegelka, S.: What can neural networks reason about? arXiv preprint arXiv:1905.13211 (2019)
61. Zhang, J., Tai, L., Boedecker, J., Burgard, W., Liu, M.: Neural SLAM. arxiv preprint 1706.09520 (2017)
62. Zhou, J., Cui, G., Zhang, Z., Yang, C., Liu, Z., Wang, L., Li, C., Sun, M.: Graph neural networks: A review of methods and applications. arXiv preprint arXiv:1812.08434 (2018)
63. Zhu, Y., Mottaghi, R., Kolve, E., Lim, J.J., Gupta, A., Fei-Fei, L., Farhadi, A.: Target-driven visual navigation in indoor scenes using deep reinforcement learning. In: 2017 IEEE international conference on robotics and automation (ICRA). pp. 3357–3364. IEEE (2017)