Optimistic Agent: Accurate Graph-Based Value Estimation for More Successful Visual Navigation

Mahdi Kazemi Moghaddam, Qi Wu, Ehsan Abbasnejad and Javen Shi
Australian Institute for Machine Learning
The University of Adelaide

Abstract

We humans can impeccably search for a target object, given its name only, even in an unseen environment. We argue that this ability is largely due to three main reasons: the incorporation of prior knowledge *(or experience), the adaptation of it to the new environment using the observed* visual cues *and most importantly* optimistically *searching without giving up early. This is currently missing in the state-of-the-art visual navigation methods based on Reinforcement Learning (RL). In this paper, we propose to use externally learned prior knowledge of the relative object locations and integrate it into our model by constructing a neural graph. In order to efficiently incorporate the graph without increasing the state-space complexity, we propose Graph-based Value Estimation (GVE) module. GVE provides a more accurate baseline for estimating the Advantage function in actor-critic RL algorithm. This results in reduced value estimation error and, consequently, convergence to a more optimal policy. Through empirical studies, we show that our agent, dubbed as the* optimistic *agent, has a more realistic estimate of the state value during a navigation episode which leads to a higher success rate. Our extensive ablation studies show the efficacy of our simple method which achieves the state-of-the-art results measured by the conventional visual navigation metrics, e.g. Success Rate (SR) and Success weighted by Path Length (SPL), in AI2THOR environment.*

1. Introduction

Human beings are capable of finding a given object in an unexplored environment, *e.g.* a room in a new house, given just a short natural language command, *e.g.* name of the object. We primarily rely on our prior knowledge of the way different objects are co-located in a specific room in addition to the new sensory input information we receive. For example, we know in any bathroom, soap bottle should be located near the basin, thus observing one (assuming lim-
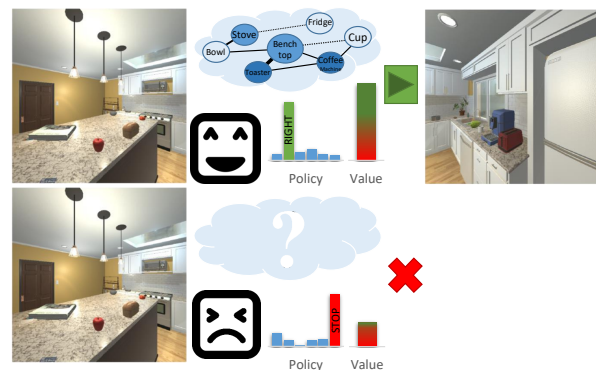

Figure 1. Our optimistic agent estimates the state value more accurately with the help of a graph representing the relationship between object locations. The more accurate value estimation helps with learning a more optimal policy that is robust to early capitulation (*e.g.* stopping before reaching the target object as a result of under-estimating the state value).

ited field of view and/or longer distance) helps us find the other more easily (*e.g.* by providing clues). Our belief, however, needs to be adjusted upon receiving new observations in a previously unseen environment. For example, the soap bottle might be misplaced or it might be in a new unseen shape or color. Finally, we do not normally give up on searching as soon as we fail to find the target in the first attempt. In other words, we optimistically continue our search as long as we receive indications from the environment that the target is reachable. It is desirable for every navigation agent to be able to utilise prior knowledge, being prepared for adapting the beliefs to a new unseen environment and being robust to early capitulation.

To endow the navigation agents with the first capability, recent works [15,17,37] have used Graph Neural Networks (GNNs) to encode the object-object relationships in a pretrained model, in the form of a knowledge graph. Their method, however, as we show in our experiments, is not efficient and cannot scale to more complex models. The major drawback of those methods is increasing the complexity of the state-space. This is crucial to avoid in sparse reward RL

set-up since it exacerbates the challenges around credit assignment and impedes the performance improvement. On the other hand, to enable the agents with the second skill (e.g. adaptation), there is an existing work [33] which incorporates meta-learning to allow the agent to quickly adapt to a new environment, using a few gradient-based weight updates. While the mentioned work shows relative success in handling the train-test distribution shift, its efficiency in adapting prior knowledge to unobserved scenes has remained unexplored. This task, as will be shown in the ablation studies, is non-trivial in RL framework. The reason is as we increase the model's complexity, efficient training becomes more challenging with conventional RL objective functions.

Finally, the last mentioned desirable capability for a navigation agent to have is a realistic (read it as optimistic) estimation of it's expected future reward. Revisiting the bathroom example again, an optimistic agent will continue to search for a soap bottle that is not found in its immediate expected location, e.g. near the basin. An unrealistic agent, however, will give up as soon as it fails to find the soap bottle near the basin without further search. In actor-critic RL a more realistic estimation of the value function will lead to a more optimal policy that is more robust to early capitulation.

To that end, we propose our Graph-based Value Estimation (GVE) module to efficiently benefit from the prior knowledge for more accurate value function estimation. The prior knowledge presents the agent with general rules about the semantic relationships of the objects and their relative location in an indoor environment. Instead of using the graph as the input to the state representation encoder, which uncontrollably increases the effective size of the state space, we use the graph to estimate the potential return [1] from the current state. This way the graph is used to better guide the policy training rather than being directly used for action taking. We present, in our experiments, that simply conditioning the policy on the prior knowledge does not lead to a more optimal policy with better performance.

Intuitively, observing a correlated object to the target that the agent is searching for has a strong indication of the final reward that can be gained, following the current policy. For instance, when searching for a book in a room, observing a desk or shelves from the distance has an indication of finding the book, hence the current state has a higher value. The association of these observations through unstructured trial and error in RL without incorporating the prior knowledge is not simply achievable; specifically, when objects are small the extracted visual features might not contain much information to guide the agent towards them.

Our GVE results in a more optimally trained critic that can associate the visual observations with its prior knowledge to better estimate the state value. This is essential in our framework where we use a variant of actor-critic RL algorithm (*e.g.* Asynchronous Advantage Actor-Critic; A3C [21], where the critic's role is to identify the actions leading to a specific reward using the temporal difference of the state values. We empirically show that GVE improves the value estimation accuracy. It is shown that the high accuracy of value estimation is essential for an optimal policy training [32], which is further confirmed in the performance of our proposed method.

In overall, we propose a simple, principled and modular approach that can be employed in conjunction with other approaches for visual navigation that also use an actor-critic RL framework. Our main contributions are:

- We introduce a simple, yet effective method that can endow an actor-critic navigation agent by improved integration of prior knowledge for navigation;

- We propose the GVE module which reduces the value estimation error in A3C RL algorithm. We empirically show the higher accuracy of our estimates which leads to a more optimal policy.

- Finally, we provide the new state-of-the-art performance results on the object-target visual navigation task in AI2THOR environment. This is achieved in the more challenging scenario of navigation using target object name only.

2. Related Work

2.1. Visual Navigation

End-to-end visual navigation has recently gained attention [1, 9, 10, 19, 20, 39] and different tasks have been introduced. Some tasks consider inputting images of the target [9, 39] while others consider the more challenging task of natural language instructions [1,2,30,33,37]. In the latter the agent has to ground language instructions on observations while performing planning and navigation. The main approaches can be divided into supervised (*e.g.* Imitation Learning) [2] and unsupervised (RL) [20] methods.

The RL-based navigation has gained extra momentum in the past few years owing to the availability of high-quality 3D simulators [2, 13, 24, 35] as well as more efficient deep RL algorithms [16, 22]. AI2THOR [13] is a photo-realistic simulator that unlike some other simulators [2] enables continuous movements of the agent. This feature, while making the navigation task more challenging is closer to real-world scenarios, hence more valuable. In this environment, it is very likely for a sub-optimal agent to stand in front of a blocked path, e.g. an obstacle, and continuously perform a failing action, e.g. move forward, until the maximum step limit is exhausted.

---

[1] Defined as the expected sum of discounted future rewards.3.1

2.2. Prior Knowledge

Adding some form of prior to machine learning models has fast-tracked the progress in many different computer vision tasks [8, 25, 27, 36]. The prior can be in the form of pre-trained weights of the network for object detection or segmentation tasks [25] or in the from of Graph Neural Networks (GNNs) representing common sense knowledge [8]. In RL framework, GNNs have been used to improve navigation models by representing topological environment maps for localisation [18] or helping with more efficient exploration [4]. Generating and incorporating scene graphs [8] is also closely related to our problem. However, here we construct our knowledge graph externally and refine the node relationships without explicitly detecting the objects in the observation. This also separates our approach from [23] and [5], where an off-the-shelf object detector is used. These methods, while improving the performance, have been explored before and can be plugged into other approaches for further improvements, including ours. The other difference of the concurrent work of Du *et al.* [5] to ours is the use of Imitation Learning (IL), which further makes it incomparable since here we only use sparse rewards and not ground-truth actions for training.

A similar work to ours is proposed by Vijay *et al.* [29] where the prior knowledge is injected into RL for navigation. The authors propose to learn various edge features encoded as one-hot vectors. The main difference, however, is that they address the navigation in a 2D fully-observable environment while in our case we deal with near-real-world task of navigation in 3D environment.

In [37] the authors use a knowledge graph to learn to find unseen objects by learning the correlations to the seen targets during the training. One of the deficiencies of their method is that it effectively increases the state space size. This is specifically important in RL framework where increasing the model's complexity can impede improving the performance, as we show in this work.

2.3. Value Function Estimation in Actor-Critic RL

One major drawback of model-free RL methods is the high sample complexity. This is specifically worsened in our 3D environment set-up where the agent has to learn the visual associations and ground the language instruction while learning the optimal policy using only sparse rewards. In order to mitigate this in actor-critic RL, some of the parameters of the policy are shared with the value estimation module which provides a stronger learning signal for the shared parameters. An accurate and stable estimate of the value, however, is necessary to reach this objective [26, 28].

In [31] the authors separate the parameters of the advantage and value functions to increase the accuracy of action-value function estimation. Inspired by that, we propose a novel value estimation module for visual navigation. This module can estimate the value of each state more accurately incorporating the object relationships encoded using a graph neural network.

3. Our Method

In this section, we first describe the task in details and then discuss our proposed method.

3.1. Preliminaries

The AI2THOR [13] task is to navigate to a target object given its name, as a word embedding vector. The navigation episode starts when the agent is spawned in a uniformly random location and orientation in one of the four rooms, *e.g.* kitchen, living room, bedroom or bathroom. The room is also sampled from a uniform distribution. Then a randomly sampled target object, from among visible objects in the scene, is presented to the agent in plain language, *e.g.* "fridge" or "soap", for example. The only accessible observation to the agent is its egocentric RGB image at each time step. The agent takes actions sampled from its policy based on the observation at each time step to locate the target object. An episode ends if either the agent stops within a defined distance of an instance of the target object or the maximum number of permitted actions is exhausted. We define the problem as a Partially-Observable Markov Decision Problem (POMDP) of $\{X, A, r, \gamma\}$. Here $\{X\}$ is the state space, the action space is A, r is the immediate reward at each step and γ is the reward discount factor. A trajectory of $(x_0, a_0, r_0, x_1, a_1, r_1...)$ is generated by taking action a_t at time t and observing the states according to the unknown dynamics of the environment $x_{t+1} \sim P(x_{t+1}|x_t, a_t)$. A reward $r_t = r(x_t, a_t, x_{t+1})$ is received from the environment at each time stop.

In this setup, the agent is trained to maximise the accumulated expected reward, $\mathbb{E}_{\tau \sim \pi}[\sum_{t=i}^{T} \gamma^t r_t]$, where τ is a trajectory sampled from the agent's policy π. The policy is approximated by a neural network, which receives the state x_t and a target object name embedding Z.

We employ A3C [21] as one of the most effective actor-critic RL algorithms to train our policy. We focus on actor-critic methods because they are popular in visual navigation due to their efficiency, ease of use and robustness. In addition, these methods bridge between gradient-based and value iteration approaches inheriting their benefits. We hypothesise that our proposed method can be applied to other actor-critic methods with minimal modifications. In our approach, the model is parameterised with $= \{\boldsymbol{\theta}, \boldsymbol{\theta}_\pi, \boldsymbol{\theta}_v\}$; $\boldsymbol{\theta}$ is the set of parameters of the backbone conditional state embedding network, $\boldsymbol{\theta}_\pi$ is the set of parameters for the policy sub-network, *a.k.a* actor, and $\boldsymbol{\theta}_v$ is the set of value sub-network's parameters, *a.k.a* critic. We use a CNN to encode the observations and Glove [11] to encode the word embed-

dings for the target. More details on the network architecture is presented in methods and implementation details.

Conventionally, the actor and critic share the network parameters. However, they are *not* tasked with the same objective: while the actor needs a minimum representation to capture the environment to take appropriate actions at each time-step, the critic needs a sufficiently holistic representation to estimate how likely it could be for the agent to achieve its goal. The critic does not need every detail of the scene and requires a global representation as opposed to that of the policy. Through the feedback the critic provides, the actor improves which is an implicit guide for the policy. With this intuition, in our proposed approach we augment the critic's sub-network with our Graph-based Value Function Estimation (GVE) module that considers and updates a knowledge graph of the object relations. GVE leads to a more accurate value estimation which then reduces the variance of the gradient samples [32]. This is because the policy is updated at each time step using gradient descent to minimise the following loss functions:

$$L_\pi(a_t|\mathbf{x}_t) \approx -\log \pi(a_t|\mathbf{x}_t; \boldsymbol{\theta}, \boldsymbol{\theta}_\pi)(r_t(a_t|\mathbf{x}_t)+$$
$$\gamma \hat{V}^\pi(\mathbf{x}_{t+1}) - \hat{V}^\pi(\mathbf{x}_t)) + \beta_H H_t(\pi) \quad (1)$$
$$L_V(x_t) = \frac{1}{2}(\hat{V}^\pi(x_t) - V(x_t))^2$$

where L_π is the policy function loss and L_V is the value function loss. Here, $\hat{V}(x_t)$ is our method's estimate of the ground truth value $V(x_t)$ at state x_t, H_t is the entropy of the policy at time step t and β_H is hyper-parameter to encourage exploration. It is shown in [32] that a more accurate estimation of the value functions leads to a lower variance of the gradients which encourages a more stable training and consequently finding a better policy.

For the adaptation, inspired by [33], we adopt Model Agnostic Meta-Learning (MAML) [6] to continuously adapt the learnt knowledge during test time. To do so, in our RL setup, we divide the training trajectories into meta-train D and meta-validation D' domains. D includes sub-trajectories of defined intervals and D' covers the complete trajectory. Then, L_{Adapt}, a loss function parameterised by ϕ is learnt to compensate for the domain shift during test. We optimise the adaptation loss and the policy loss L_π according to the following update rules:

$$\boldsymbol{\theta}_{\text{total}}^i \leftarrow \boldsymbol{\theta}_{\text{total}} - \alpha \nabla_{\boldsymbol{\theta}_{\text{total}}} L_{Adapt}^{\tau \sim D}(x_t, \pi(x_t); \phi)$$
$$\boldsymbol{\theta}_{\text{total}} \leftarrow \boldsymbol{\theta}_{\text{total}}^i - \beta_1 \nabla_{\boldsymbol{\theta}_{\text{total}}^i} L_\pi^{\tau \sim D'}(x_t, \pi(x_t)) \quad (2)$$
$$\phi \leftarrow \phi - \beta_2 \nabla_\phi L_\pi^{\tau \sim D'}(x_t, \pi(x_t))$$

We define $\boldsymbol{\theta}_{\text{total}} = \{\boldsymbol{\theta}, \boldsymbol{\theta}_\pi\}$ in the above update rules for readability, α, β_1 and β_2 are the learning rate hyper-parameters and θ_{total}^i is the set of intermediate weights after the adaptation before the final update. In practice we can have multiple intermediate updates i before the final update.

3.2. Prior Knowledge Graph

Our GVE module relies on a prior knowledge graph $G(V, E)$ which contains the information about the co-location of the objects in a scene. Intuitively, if the joint probability of observing object i and j together in a single view is high in an environment, then observing one should provide a strong cue for finding the other. Based on this intuition, inspired by the work of [37], we define an object co-location graph. The set of nodes V includes all the objects visible in the environment (*e.g.* whether used for navigation target or not) represented by their features. Each node feature, $\mathbf{v}_i \in \mathbb{R}^d$, encodes the concatenation of the RGB observation features extracted using a CNN and the semantic vector embedding of the object, extracted using a word embedding model. The edges show the co-location of the objects; that is, e_{ij} becomes incresingly closer to one during the training if and only if the objects i and j often appear in the same egocentric view of the agent, simultaneously.

We have multiple separate graphs encoding the objects visible in different room types. This is one of the novelties of our method which is, particularly, important since similar objects might appear in different rooms while their co-locating objects might be different. This will allow for having different edge values for our graph in each room. For instance, the object "bowl" might appear both in the "kitchen" and the "living room". It might be observed near the "cabinets" in the "kitchen" but close to "sofa" in the "living room". Therefore, our adjacency matrix, $A \in \mathbb{R}^{n \times n \times C}$, is a three-dimensional tensor where each channel C encodes the knowledge specific to a room type. We perform this by initialising the edges between objects that do not appear in each specific room type with zero. The graph separation enables the agent to mainly attend to one of the graph channels in each scene and avoid distraction. This way, more scene-specific knowledge can be encoded.

Inspired by [37], we also initialise the graph's binary edge weights using the co-occurrence of the objects in images acquired from the Visual Genome dataset [14]. However, we learn more accurate edge weights along with node features during the training using the attention mechanism, as proposed in Graph Transformer Network [38].

3.3. Graph-based Value Function Estimation in Actor-Critic

It is proven that policy gradient algorithms in RL have high variance of gradient estimation and different solutions have been proposed to address this [7, 34]. The issue is mainly due to the difficulty of credit assignment which is exacerbated where the reward is sparse. In actor-critic methods the variance is reduced by using the boot-strapped estimates of the state-value function as a baseline. It is also known that a more accurate estimate of the value function
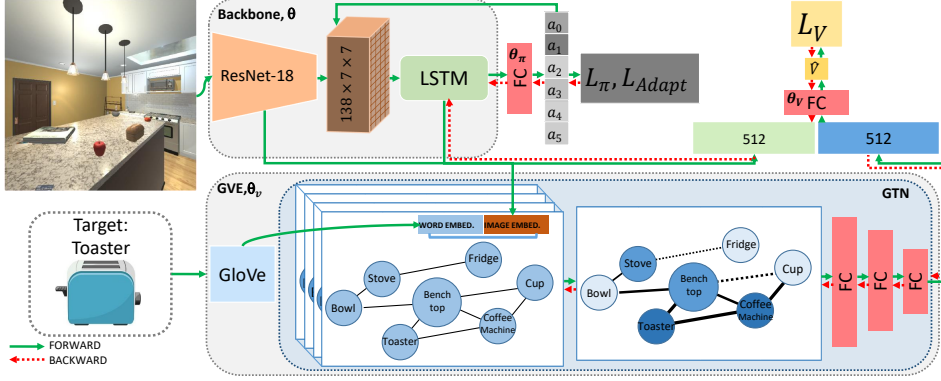
Figure 2. Overview of our method. Our GVE module augments the A3C backbone and is trained using the supervised loss L_V at each time step of the trajectory.

leads to a more optimally converged policy [32].

Accurate estimation of the state value in visual navigation is significantly challenging, particularly when searching for a target object. This is mainly due to the lack of global context at each time step where the agent can only observe a limited view of the scene. The context is conventionally encoded using the recurrent neural networks which are prone to forgetting and challenging to train. We propose our Graph-based Value Estimation (GVE) module to address this problem. Using GVE we encode the global context of object locations into our model so the agent has a better understanding of its position in the scene with respect to other objects. This will help the agent more realistically estimate the state value. Intuitively, there's a correlation between the objects present in each time-step's viusal input and the target object to navigate to. For example, when navigating to a smaller object like "book" in a room, the agent may look for larger indicator objects like desk or shelf that guide it towards the target. This means that the agent should be *optimistic* about finding the target object by observing a correlated object based on its prior knowledge. In other words, a state where a correlated object is visible should still have a high estimated value, close to the value of a state where the target is visible.

In order to encode our prior knowledge graph for value estimation, we propose to use Graph Transformer Networks (GTN) [38]. Using GTN our agent is able to learn new edges and update the edge weights. This way, depending on the scene, we are able to extract features from the graph that help with more efficient navigation. In order to update the original edge weights we first learn a soft weighted average of the edges across multiple channels of the input adjacency matrix:

$$H_i^l = \text{softmax}(\boldsymbol{W_i^l})A \qquad (3)$$

where $W_i^l \in \mathbb{R}^{1 \times 1 \times C}$ is a weight matrix over the channels. H_i will be the concatenation of multiple H_i^l matrices to learn multiple new mappings of the original adjacency matrix. In practice we can learn M different H_i and the final new adjacency matrix is then defined as the matrix multiplication of those which can encode the common paths:

$$A_{NEW} = H_1 H_2 ... H_M \qquad (4)$$

Finally, in order to encode the node features using the new adjacency matrix A_{NEW}, we use graph convolutional layers defined as:

$$Q = \sigma(\tilde{E}_i^{-1} \tilde{A}_{NEW} \boldsymbol{W_N} N) \qquad (5)$$

where Q is the final embedding of the graph, $N \in \mathbb{R}^{n \times d}$ is the input node features, $\boldsymbol{W_N}$ is the weights for node embedding, $\tilde{A}_{NEW} = A_{NEW} + I$ is the augmented learned adjacency matrix with self-connections using identity matrix I and $\tilde{E}^{-1}$ is the inverse degree matrix for $\tilde{A}_{NEW}$. In practice we can learn multiple convolutional layers embedding different node features using the new adjacency matrix. Therefore, the output graph representation vector Q is the result of both node and edge operations dynamically learnt during training.

In order to incorporate the graph's encoding into our GVE, we partially separate the parameters of the critic sub-network, $\boldsymbol{\theta}_v$, from the actor sub-network $\boldsymbol{\theta}_\pi$. Therefore, in our method we estimate the state-value function according to:

$$\hat{V}(x_t) = \boldsymbol{W_1}Q + \boldsymbol{W_2}F(x_t, Z; \boldsymbol{\theta}) \qquad (6)$$

$\boldsymbol{W_1}$ and $\boldsymbol{W_2}$ are aggregation parameters for the final value estimation which are also trained along the other parameters of the network and $\boldsymbol{\theta}$ is the set of parameters for F, the state encoding backbone network.

As can be seen in equation 6, our method decomposes the expected future reward into two components: one, estimated mainly conditioned on the state as encoded by the encoding network parameterised with $\boldsymbol{\theta}$; and the other component, Q, estimated by the graph encoding network which represents a correlations between the target object and the

objects that are visible in the current state x_t. Therefore, the agent can include the expectation of the objects it might observe in the future states in order to estimate the value of current state more accurately. In other words, the agent has a more global understanding of the environment.

4. Experiments

4.1. Experimental Setup

We choose AI2THOR environment for our experiments. This simulator consists of photo-realistic indoor environments (e.g. houses) categorised into four different room types: kitchen, bedroom, bathroom and living room. In order for fair comparison, we follow the same experimental setup as our main baseline [33]. In this setup, 20 different scenes of each room type are used for training; 5 scenes for each as validation and 5 for test. Following the recent conventions in visual navigation tasks, we measure the performance of our method based on Success Rate (SR) and Success weighted by Path Length (SPL).

4.2. Implementation Details

Our actor-critic network comprises of a LSTM with 512 hidden states and two fully-connected layers one for actor and the other for the critic. The actor outputs a 6-dimensional distribution $\pi(x_t)$ over actions using a Softmax function while the critic predicts a single scalar value. The critic sub-network also receives the GTN encoding as a vector of size 512 besides the 512-dimensional LSTM encoding. The concatenation of these two feature vectors is mapped to the value using a fully connected layer.

We use Glove [11] to generate 300-dimensional semantic word embedding of the objects and a pre-trained ResNet-18 to extract the features from the 300×300 image inputs. We concatenate these two to feed to our state encoder, LSTM. The overview of the architecture can be seen in figure 2

Our prior knowledge graph is pre-trained on Visual Genome dataset [14]. The edge weights are set to one where the objects co-occur in a scene at least three times, and zero otherwise. There are 89 nodes in each channel of the graph and 5 channels in total; 4 layers dedicated to the objects in each room type and one self-connections layer for regularisation. We train a two layer adjacency matrix using GTN. The input to the graph, as node features, is a 1024-dimensional vector. This vector is the concatenation of 512-dimensional observation features, extracted using ResNet-18, with 512-dimensional Glove [11] embedding of the object name. The Glove embedding is mapped from 300 to 512 using a fully connected layer. The GTN features are mapped into a 512-dimensional vector using a fully connected layer.

In order to train our model, we use Pytorch framework. We use SGD optimizer for meta-train weight updates and
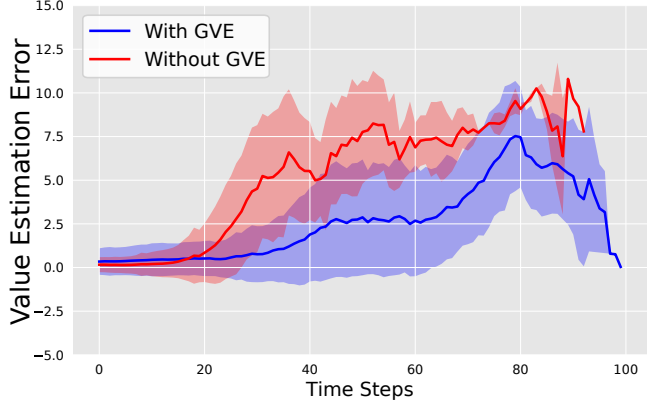

Figure 3. Value estimation error on test set. Our model has more accurate estimates of the state value function (lower error) which leads to better performance.

Adam [12] for the meta-test. As for the reward, we use a constant value of 5 for reaching the target and a step penalty of -0.01 for each single step. The maximum number of steps is capped at 50 during training and 200 during testing. We train all our methods until convergence with the maximum seven million episodes, whichever occurs first.

4.3. Baseline and SOTA Comparison

In order to better show the contribution our method we first compare it with a few different state-of-the-art (SOTA) and baselines, shown in Table 1. Firstly, we compare with the previous SOTA introduced by Wortsman *et al*. [33], abbreviated as **A3C+MAML**. It uses MAML [6] to learn a loss function approximated by an instance of Temporal Convolutional Networks [3], over training episodes. Secondly, a related work [37] uses a fixed knowledge graph structure to encode object relationships as part of the state space, abbreviated as **A3C+Graph**. In our approach, we benefit from the graph information more efficiently using our proposed GVE module. In addition to that, our graph structure and embedding architecture is also different, enabling more accurate estimation of the value function.

Furthermore, to highlight the challenging navigation scenarios that our method is tackling, we compare our results with some trivial baselines. One simple solution is a **Random** agent for which the policy is to uniformly sample an action at all times. Another trivial baseline is shown as **A3C** in Table 1. This method is the result of dismantling our GVE and adaptation modules from the overall framework. Therefore, it acts as the simplest RL-based agent that shares the same backbone model with our proposed method.

4.4. Results and Ablation Studies

In this section we seek to answer a few principal questions with regards to our proposed method that shed more light on its strengths as well as weaknesses.

Table 1. Quantitative comparison of our results with the baselines. Our approach improves all the baselines in all the four evaluation metrics, conventionally used in the previous state-of-the-art methods.

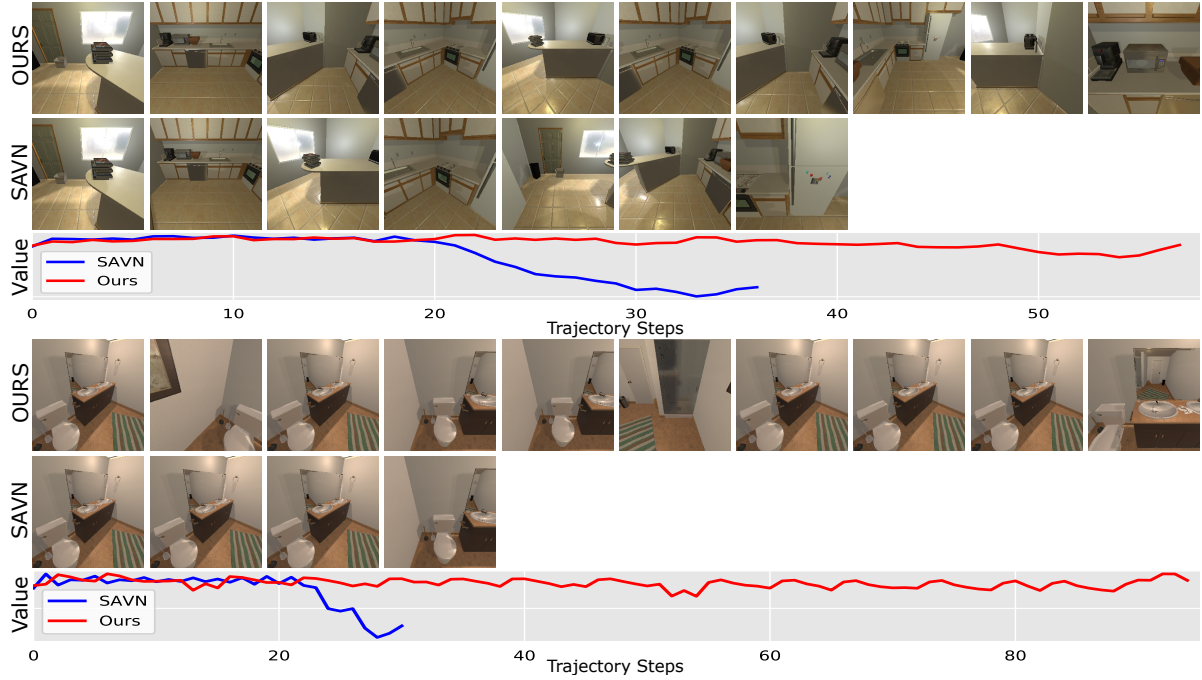| Method	SPL	Success	SPL >5	Success >5
Random	3.64	8.01	0.1	0.28
A3C	14.68 ±1.8	33.04 ±3.5	11.69 ±1.9	21.44 ±3.0
A3C+Graph [37]	15.47 ±1.1	35.13 ±1.3	11.37 ±1.6	22.25 ±2.7
A3C+MAML(SAVN) [33]	16.15 ±0.5	40.86 ±1.2	13.91 ±0.5	28.70 ±1.5
A3C+Graph+GVE-ours	16.02	38.22	13.23	27.47
A3C+Graph+MAML-SS-ours (w/o. GVE)	15.13	38.8	13.68	29.64
A3C+Graph+MAML-Action-ours (w/o. GVE)	13.88	43.3	12.93	33.53
A3C+Graph+MAML+GVE-ours	**17.27** ±0.3	**43.8** ±1.1	**15.39** ±0.2	**33.68** ±0.9

Figure 4. Qualitative comparison of two sample trajectories between our method and the baseline. **Top**: target object: "bowl" in the kitchen. Our agent observes the bench-top and continuously predicts a high value (*being optimistic*) until finding the target. The baseline agent, however, has a less accurate understanding of the scene and *gives up* after a while not detecting the target. **Bottom**: target object: "soap bottle" in the bathroom. Our agent observes the basin and constantly predicts a high state value even though the soap bottle is small to be detected in the visual features. The baseline gives up after a few failed actions of moving forward.

- *How does GVE improve value estimation error?*

In order to respond to this question, we analyse the estimated values over the trajectories using our GVE in comparison to the baseline method. The estimation error over 1000 trajectories collected in the test scenes (unseen during training) is provided in Figure 3. The figure shows average and standard deviation of L2 distance per time step between the ground-truth value and the predictions. The red line shows the error for the baseline while the blue line shows the improvement as a results of our GVE. Our method effectively addresses the forgetting problem associated with LSTMs by estimating the values for longer trajectories more accurately.

- *Is GVE the optimal way to incorporate prior knowledge?*

In order to show the effectiveness of our graph-based value estimation, we compare our final method with two variants. First variant, termed as **A3C+Graph+MAML-SS** is to simply add our graph as part of the state space observation to the backbone network. As can be seen in Table 1 this method improves SR but not the SPL compared to **A3C+Graph**; however, compared to **A3C+MAML** in degrades the performance. We conjugate that the reason is by adding the complex graph to the backbone network, the size of the state space is increased by orders of magnitude which impedes learning an effective state encoding under

Table 2. *Ours-LG* is the variant of our model where the image features are removed from the graph node features. We can see the graph is highly reliant on the observations to learn the relationships.

Method	SPL	Success
Ours-LG	12.84	42.4
Ours-GVE	17.27	43.8

Table 3. *Ours-RandomGraph* is the variant of our model where the edge weights are initialised randomly. The results on this table show the contribution of prior knowledge in our framework.

Method	SPL	Success
Ours-RandomGraph	15.32	40.5
Ours-GVE	17.27	43.8

sparse reward constraint.

In another variant, termed as **A3C+Graph+MAML-Action**, we directly incorporate the graph into the policy for better decision making. Therefore, the policy will take actions conditioned upon the graph encoding. This approach can be observed as a weighted ensemble of policies representing different distributions. This leads to an improvement on SR but a significant degradation of SPL. We hypothesise that this can help the policy on better detecting the target object and thus on-time stopping which increases the SR; however, it cannot help with the optimality of the final policy. This further proves the effectiveness of our proposed GVE.

- What is the contribution of prior knowledge for value estimation?

In order to show the significant contribution of prior knowledge in our current framework we train our final model with randomly initialised edge weights in the graph. As can be seen in 3, adding the graph with random edge weights does not improve the results compared to our baseline methods. This further confirms that our GVE module is reliant on the prior knowledge for more accurate value estimation.

- *What is the quantitative improvement with respect to the baselines and previous state-of-the-art methods?*

As can be seen in Table 1, our final model **A3C+Graph+MAML+GVE** improves the previous SOTA **A3C+MAML** by almost 3% on SR and more than 1% on SPL. This shows that our GVE helps the agent to find more targets in smaller number of steps, as a result of having a more optimal policy. Our method is, particularly, more effective on longer trajectories where it improves the baseline by almost 5% on SR and almost 2% on SPL. This, again, confirms the effectiveness of our GVE which can provide enough global context disentangled from the length of the trajectory.

Table 4. Detailed comparison with the baseline method; SPL/Success rate are reported per room type. We can see that our method is general enough that improves the performance in 3/4 of the room types, with marginal performance on 1/4.

Method	Bathroom	Bedroom	Kitchen	Living
A3C+MAML	28.49/69.6	**8.65/29.2**	17.8/43.6	7.71/21.6
Ours-GVE	**31.03/75.6**	8.06/27.6	**17.93/45.6**	**9.41/25.2**

- *Is providing visual features to the graph nodes helpful?*

In order to show the integrity of current design, we show the effect of removing observation (egocentric image) features from the node feature matrix. Thus, the graph will reduce to a fixed correlations among the objects, given in language embedding only (hence the name Language Graph, LG). It can also be observed as a sub-network for the value estimation to store value decomposition information without considering the observational correlations. As can be seen in Table 2, the performance of the model **Ours-LG** is significantly lower, particularly in SPL. This shows that the graph's contribution is not due to simply adding more trainable parameters.

- *How is our more optimal policy performing qualitatively?*

Finally, we provide qualitative comparison of the performance of our method compared to previous SOTA , *e.g.* **A3C+MAML** [33]. As is shown in Figure 4 the agent is navigating towards an instance of "microwave" in a kitchen scene. Our agent is able to find the target while constantly predicting a high value for the states due to observing the related objects like the "benchtop". The baseline, however, is after a few steps predicts low state values and *gives up* without reaching to the target.

For more detailed comparison, we also provide performance results per room type, in Table 4. Additional trajectory visualisations can be found in supplementary material.

5. Conclusion

In this paper we present a simple yet effective method to improve the performance of the actor-critic RL algorithm for visual navigation. Our method improves the state value function estimation accuracy by incorporating object co-location information in the form of a neural graph. Actor-critic RL is heavily reliant on the accuracy of the state value estimation to converge to the optimal policy. This is mainly because accurate value estimation helps with correct identification of the actions along the trajectory that contribute the most to the final reward. Through extensive empirical studies we show that our agent is realistically more optimistic (*e.g.* accurately predicts higher state values). This leads to successful navigation towards the target object when the baseline agents usually give up.

References

[1] Peter Anderson, Angel Chang, Devendra Singh Chaplot, Alexey Dosovitskiy, Saurabh Gupta, Vladlen Koltun, Jana Kosecka, Jitendra Malik, Roozbeh Mottaghi, Manolis Savva, et al. On evaluation of embodied navigation agents. *arXiv preprint arXiv:1807.06757*, 2018.

[2] Peter Anderson, Qi Wu, Damien Teney, Jake Bruce, Mark Johnson, Niko Sunderhauf, Ian Reid, Stephen Gould, and Anton van den Hengel. Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Jun 2018.

[3] Shaojie Bai, J. Zico Kolter, and Vladlen Koltun. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling, 2018.

[4] Hanjun Dai, Yujia Li, Chenglong Wang, Rishabh Singh, Po-Sen Huang, and Pushmeet Kohli. Learning transferable graph exploration, 2019.

[5] Heming Du, Xin Yu, and Liang Zheng. Learning object relation graph and tentative policy for visual navigation, 2020.

[6] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks, 2017.

[7] Evan Greensmith, Peter L Bartlett, and Jonathan Baxter. Variance reduction techniques for gradient estimates in reinforcement learning. *Journal of Machine Learning Research*, 5(Nov):1471–1530, 2004.

[8] Jiuxiang Gu, Handong Zhao, Zhe Lin, Sheng Li, Jianfei Cai, and Mingyang Ling. Scene graph generation with external knowledge and image reconstruction. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun 2019.

[9] Saurabh Gupta, James Davidson, Sergey Levine, Rahul Sukthankar, and Jitendra Malik. Cognitive mapping and planning for visual navigation. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jul 2017.

[10] Saurabh Gupta, David Fouhey, Sergey Levine, and Jitendra Malik. Unifying map and landmark based representations for visual navigation. *arXiv preprint arXiv:1712.08125*, 2017.

[11] RichardSocher JeffreyPennington and ChristopherD Manning. Glove: Global vectors for word representation. Citeseer.

[12] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[13] Eric Kolve, Roozbeh Mottaghi, Winson Han, Eli VanderBilt, Luca Weihs, Alvaro Herrasti, Daniel Gordon, Yuke Zhu, Abhinav Gupta, and Ali Farhadi. Ai2-thor: An interactive 3d environment for visual ai, 2017.

[14] Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David A. Shamma, and et al. Visual genome: Connecting language and vision using crowdsourced dense image annotations. *International Journal of Computer Vision*, 123(1):32–73, Feb 2017.

[15] Ruiyu Li, Makarand Tapaswi, Renjie Liao, Jiaya Jia, Raquel Urtasun, and Sanja Fidler. Situation recognition with graph neural networks. *2017 IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[16] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

[17] Kenneth Marino, Ruslan Salakhutdinov, and Abhinav Gupta. The more you know: Using knowledge graphs for image classification. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jul 2017.

[18] Xiangyun Meng, Nathan Ratliff, Yu Xiang, and Dieter Fox. Scaling local control to large-scale topological navigation, 2019.

[19] Piotr Mirowski, Matt Grimes, Mateusz Malinowski, Karl Moritz Hermann, Keith Anderson, Denis Teplyashin, Karen Simonyan, Andrew Zisserman, Raia Hadsell, et al. Learning to navigate in cities without a map. In *Advances in Neural Information Processing Systems*, pages 2419–2430, 2018.

[20] Piotr Mirowski, Razvan Pascanu, Fabio Viola, Hubert Soyer, Andrew J Ballard, Andrea Banino, Misha Denil, Ross Goroshin, Laurent Sifre, Koray Kavukcuoglu, et al. Learning to navigate in complex environments. *arXiv preprint arXiv:1611.03673*, 2016.

[21] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In Maria Florina Balcan and Kilian Q. Weinberger, editors, *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 1928–1937, New York, New York, USA, 20–22 Jun 2016. PMLR.

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

[23] Arsalan Mousavian, Alexander Toshev, Marek Fiser, Jana Kosecka, Ayzaan Wahid, and James Davidson. Visual representations for semantic target driven navigation. *2019 International Conference on Robotics and Automation (ICRA)*, May 2019.

[24] Manolis Savva, Abhishek Kadian, Oleksandr Maksymets, Yili Zhao, Erik Wijmans, Bhavana Jain, Julian Straub, Jia Liu, Vladlen Koltun, Jitendra Malik, Devi Parikh, and Dhruv Batra. Habitat: A platform for embodied ai research, 2019.

[25] Alexander Sax, Bradley Emi, Amir R. Zamir, Leonidas Guibas, Silvio Savarese, and Jitendra Malik. Mid-level visual representations improve generalization and sample efficiency for learning visuomotor policies, 2018.

[26] John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation, 2015.

[27] William Shen, Danfei Xu, Yuke Zhu, Li Fei-Fei, Leonidas Guibas, and Silvio Savarese. Situational fusion of visual representation for visual navigation. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, Oct 2019.

[28] Richard S Sutton, David A McAllester, Satinder P Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Advances in neural information processing systems*, pages 1057–1063, 2000.

[29] Varun Kumar Vijay, Abhinav Ganesh, Hanlin Tang, and Arjun Bansal. Generalization to novel objects using prior relational knowledge, 2019.

[30] Xin Wang, Wenhan Xiong, Hongmin Wang, and William Yang Wang. Look before you leap: Bridging model-free and model-based reinforcement learning for planned-ahead vision-and-language navigation. *Lecture Notes in Computer Science*, page 38–55, 2018.

[31] Ziyu Wang, Tom Schaul, Matteo Hessel, Hado van Hasselt, Marc Lanctot, and Nando de Freitas. Dueling network architectures for deep reinforcement learning, 2015.

[32] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992.

[33] Mitchell Wortsman, Kiana Ehsani, Mohammad Rastegari, Ali Farhadi, and Roozbeh Mottaghi. Learning to learn how to learn: Self-adaptive visual navigation using meta-learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6750–6759, 2019.

[34] Cathy Wu, Aravind Rajeswaran, Yan Duan, Vikash Kumar, Alexandre M Bayen, Sham Kakade, Igor Mordatch, and Pieter Abbeel. Variance reduction for policy gradient with action-dependent factorized baselines. *arXiv preprint arXiv:1803.07246*, 2018.

[35] Fei Xia, Amir R Zamir, Zhiyang He, Alexander Sax, Jitendra Malik, and Silvio Savarese. Gibson env: Real-world perception for embodied agents. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9068–9079, 2018.

[36] Kelvin Xu, Ellis Ratner, Anca Dragan, Sergey Levine, and Chelsea Finn. Learning a prior over intent via meta-inverse reinforcement learning. volume 97 of *Proceedings of Machine Learning Research*, pages 6952–6962, Long Beach, California, USA, 09–15 Jun 2019. PMLR.

[37] Wei Yang, Xiaolong Wang, Ali Farhadi, Abhinav Gupta, and Roozbeh Mottaghi. Visual semantic navigation using scene priors. *arXiv preprint arXiv:1810.06543*, 2018.

[38] Seongjun Yun, Minbyul Jeong, Raehyun Kim, Jaewoo Kang, and Hyunwoo J. Kim. Graph transformer networks, 2019.

[39] Yuke Zhu, Roozbeh Mottaghi, Eric Kolve, Joseph J. Lim, Abhinav Gupta, Li Fei-Fei, and Ali Farhadi. Target-driven visual navigation in indoor scenes using deep reinforcement learning. *2017 IEEE International Conference on Robotics and Automation (ICRA)*, May 2017.