ForeSI: Success-Aware Visual Navigation Agent

Mahdi Kazemi Moghaddam, Ehsan Abbasnejad, Qi Wu, Javen Qinfeng shi and Anton Van Den Hengel
The Australian Institute for Machine Learning
The University of Adelaide
mahdi.kazemimoghaddam, ehsan.abbasnejad, qi.wu01, javen.shi and anton.vandenhengel@adelaide.edu.au

Abstract

In this work, we present a method to improve the efficiency and robustness of the previous model-free Reinforcement Learning (RL) algorithms for the task of object-goal visual navigation. Despite achieving state-of-the-art results, one of the major drawbacks of those approaches is the lack of a forward model that informs the agent about the potential consequences of its actions, i.e.*, being model-free. In this work, we augment the model-free RL with such a forward model that can predict a representation of a future state, from the beginning of a navigation episode, if the episode were to be successful. Furthermore, in order for efficient training, we develop an algorithm to integrate a replay buffer into the model-free RL that alternates between training the policy and the forward model. We call our agent ForeSI; ForeSI is trained to imagine a future latent state that leads to success. By explicitly imagining such a state, during the navigation, our agent is able to take better actions leading to two main advantages: first, in the absence of an object detector, ForeSI presents a more robust policy,* i.e.*, it leads to about 5% absolute improvement on the Success Rate (SR); second, when combined with an off-the-shelf object detector to help better distinguish the target object, our method leads to about 3% absolute improvement on the SR and about 2% absolute improvement on Success weighted by inverse Path Length (SPL),* i.e.*, presents higher efficiency.*

1. Introduction

Object-goal visual navigation is the task where an agent needs to navigate towards an instance of a given target object in a previously unseen environment. This is an inherently challenging problem since the agent needs to find the shortest possible sequence of actions to first find a target object (*i.e*. to intelligently explore) and then navigate towards it (*i.e*. to plan and act accordingly while avoiding obstacles) in a visually complex 3D environment [6, 20, 32, 38, 40]. The task is further complicated when at each time step the agent only receives limited field-of-view visual inputs only.

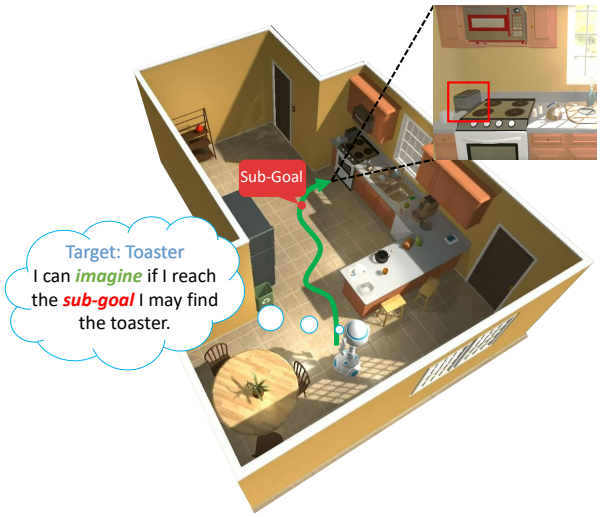

Figure 1. Enabling an agent to imagine, *i.e*., to predict, states on the path to success improves its ability to carry out complex tasks, particularly in unseen environments. As opposed to the conventional approaches, our ForeSI agent takes actions not only based on the current state, but also a prediction of a successful future, to achieve its goal.

This means the agent will need to learn simultaneously to (1) build a good state representation that enables the agent to localise itself as well as the target (if observed at some point) and (2) efficiently use that state representation to (implicitly) plan and navigate towards the target while avoiding the obstacles. In a navigation episode, from every starting point, there are typically multiple action sequences (*i.e*. trajectories) that could lead to success, let alone the sequences that might fail. Learning to select the right action at each time step to create a trajectory that leads to the specified target object is the primary challenge.

To tackle this problem, model-free RL algorithms are typically used to train a single policy to perform all the aforementioned tasks implicitly by mapping the input visual observations into the actions [3, 7, 8, 25, 39]. Explicitly incorporating the transition in the environment to predict the potential outcome of the actions, hailed model-based RL, is also developed on other tasks and environ-

ments [10, 11, 13, 16, 42]. Those methods, however, are generally harder to train, especially in a 3D rich environment, since every state transition in the environment has to be accurately modelled. Therefore, their successful application has been limited to much simpler benchmarks such as Atari [13, 16, 33], Deepmind Control Suite [11, 12] or robotic arm tasks [27, 29, 43]; unlike those problems, in visual navigation, the testing environment is unseen, visually much more complex and at times significantly different from the observed training environments.

To mitigate the above mentioned issues we propose our *Foresight Sub-goal Imaginator* (ForeSI) agent. Intuitively, as shown in Figure 1, ForeSI enables the agent to have a foresight of a future sub-goal state through which it is more likely to successfully achieve its goal. By explicitly incorporating the sub-goal information into the policy, the agent can take better actions even when the target is not in the field of view of it.

In particular, ForeSI helps with the navigation in two main ways: firstly, it provides the agent with an imagined (*i.e*. predicted) representation of a sub-goal state that will help with successful task completion, *i.e*. stopping at the right location; our empirical results in the absence of an object detector support this hypothesis; secondly, it helps the agent to constantly remember the sub-goal state to navigate to even if that state is temporarily out of the field-of-view of it; *i.e*. it improves the navigation efficiency. In summary, in this work, our primary contributions are as follows:

1. We propose a method to enable visual navigation agents to generate foresight of the states on the path to success, *i.e*. sub-goal states, conditioned on the first state and the target object.
2. We propose an attention mechanism for state-value estimation that helps to identify the sub-goal state used to train our sub-goal generation module.
3. We propose an algorithm to efficiently integrate our method into model-free RL which re-utilises the past trajectories in hindsight via incorporating a replay buffer; thus ForeSI does not require extra data collection.
4. We show the effectiveness of our method in improving visual navigation performance when added to multiple different baselines with different perception methods and environment setups on AI2THOR [20].

2. Related Work

Visual Navigation Research on visual navigation has attracted higher attention in recent years mainly due to the availability of photo and sensor realistic 3D simulators [6, 20, 32, 38, 40]. Various methods [4, 5, 8, 21, 39, 41, 44] have been proposed. Some of them use Imitation Learning (IL) as an approach to initialise a better policy [7, 8] when expert demonstrations (*i.e*. state-action pairs) are available; however, those demonstrations are expensive to collect in real-world and methods trained using IL only typically suffer from lack of generalisation. Therefore, model-free RL algorithms are at the heart of most of the previous approaches [4, 5, 39, 41, 44]. In [39] the authors propose to use meta-learning [9] to enable test-time adaptation while in [7] and [8] the authors focus on improving the perception; in [7] the authors develop a graph neural network that uses an off-the-shelf object detector [30] to exploit object co-occurrence relationship. In a more recent work, Du *et al*. [8] propose to further improve the perception by developing a Transformer [36] model to learn the object relations more efficiently. In contrast to those works, the main focus of our work is to improve the core RL algorithm by explicitly modelling the successful future of trajectories and integrating that into the policy for more robust and efficient decision making; we show our method works well along with multiple different perception modules [7, 22, 39, 41]. For the first time, we augment model-free visual navigation methods with a forward model that works well. Therefore, in what follows we also review closely related methods developed on other problems/ tasks.

Model-Based RL ForeSI uses a forward model to generate future state representations, thus is closely related to model-based RL [13, 16, 17, 33, 43]. In [37] the authors train a recurrent model of the environment dynamics that can be used for planning by unrolling an episode in imagination. Similarly, in a more recent method [11, 13], the authors train the policy purely on imagined episodes. In all of those methods, the environment model is a recurrent reconstruction-based model inspired by [10] which models every state transition. Those methods have only recently surpassed the model-free performance only on simpler toy environments [16]; their generalisation to realistic visual navigation in a 3D environment has not yet been explored which means our method is the first step towards enabling model-based RL in visual navigation. Moreover, in contrast to those methods, we enable longer-term single-step forward modelling which boosts the navigation performance in unseen environments.

Experience Replay Modifying the distribution of episodes in a replay buffer, known as hindsight experience replay, has been explored in other applications [2, 31, 35]. The core idea of those approaches is to modify the distribution of goal states in the replay buffer (*i.e*. hindsight) to train a more generalisable policy able to reach arbitrary goals provided during test time. Differently from them, here we introduce a method that is trained in hindsight and tested in foresight to generate successful (sub-)goal states for visual navigation.

Goal-Conditioned RL Our method is also closely related to goal-conditioned RL where the policy is trained while conditioned on a given goal state. In [28] the authors use

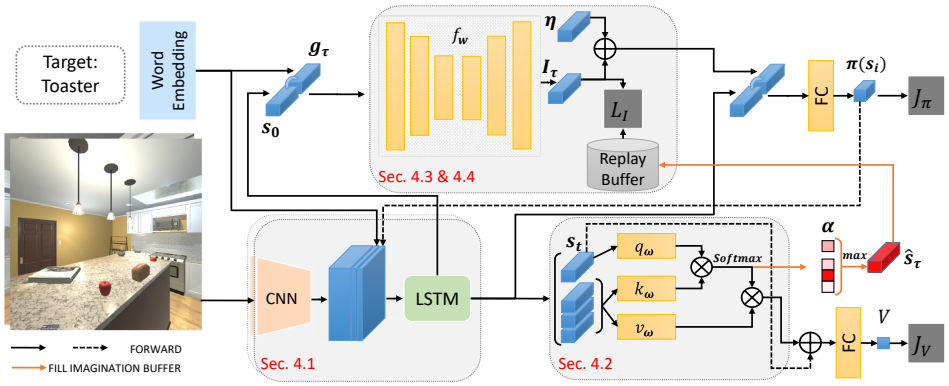

Figure 2. We develop an attention mechanism to identify the sub-goal state that minimises the critic error (in hindsight, at the end of an episode) and train our sub-goal generator to predict that state (in foresight, in the beginning of the episode). Our training algorithm efficiently integrates ForeSI into actor-critic RL while avoiding extra data collection.

combinatorial optimisation on the states encoded using a Variational Auto-Encoder (VAE) [19] to find a set of reachable sub-goals along a past trajectory. They use the selected sub-goals to optimise the policy using Q-learning [24], an off-policy RL algorithm. In [26] the authors use self-imagined goals to train the agent to reach arbitrary goals. Most recently, in [18] the authors use a VAE to learn to generate goals during test time. While those methods are related to ours, there are two fundamental differences: (1) unlike ours, the application of those methods has been limited to simpler environments where there is little to no difference between training and test environments; (2) unlike those most of those methods we use the selected sub-goals during the training to learn to generate them in unseen test environments, for the success.

3. Background

In visual navigation, the objective for the agent starting from a random initial location is to take a sequence of actions (collectively, a trajectory) to reach a given target object using only observed ego-centric RGB visual inputs. A navigation episode ends if either the agent takes the "STOP" action or the maximum number of permitted actions is exhausted. A trajectory is considered successful if the agent stops within a defined circular proximity of any instance of the target object. This problem is conventionally defined as a Partially-Observable Markov Decision Process (POMDP) denoted by the tuple $\{O, S, A, \mathcal{G}, \mathcal{P}, r, \gamma\}$ [22]. Here, O is the space of visual observations, S is the state space as encoded and observed internally by the agent, A is the action space, $\mathcal{G}$ is the set of target objects given by the environment, $\mathcal{P} := p(\mathbf{s}_t|\mathbf{s}_{t-1}, a_{t-1})$ is the transition function or environment dynamics model (unknown) for state $\mathbf{s}_t \in S$, and r is the reward function and γ is the reward discount factor.

Formally, a trajectory τ of length $T+1$ consists of a tuple $(\mathbf{s}_0, a_0, r_0; \mathbf{s}_1, a_1, r_1; ..., \mathbf{s}_T, a_T, r_T)$ that is generated by taking action a_t at time t and observing the next state according to the dynamics of the environment $\mathbf{s}_{t+1} \sim p(\mathbf{s}_{t+1} \,|\, \mathbf{s}_t, a_t)$. A reward $r_t = r(\mathbf{s}_t, a_t, \mathbf{s}_{t+1})$ is received from the environment at each time step, which is conventionally [7, 8, 39, 41, 44] defined as a large positive number for a successful trajectory and a negative step penalty otherwise, to encourage success in lowest possible number of actions.

A common approach is to use actor-critic RL methods for learning the optimal policy[1] [7,39,41,44] for navigation $\pi_\theta(a_t \,|\, \mathbf{s}_t, \mathbf{g}_\tau, \boldsymbol{\theta})$ to choose action a_t at time t conditioned on an embedding of the target object's name $\mathbf{g}_\tau$ (e.g. using GloVe embedding [15]) for a given environment (*i.e.* a room in AI2Thor) E. We use $\boldsymbol{\theta}$ to denote the set of all the parameters of the RL policy. Learning involves minimising $\mathcal{J}_\pi(a_t \,|\, \mathbf{s}_t, \boldsymbol{\theta})$ the negative of expected advantage function while minimising the difference of the estimated value function and the true return $\mathcal{J}_V(\mathbf{s}_t, \boldsymbol{\theta})$ where we have:

$$\mathcal{J}_\pi(a_t \,|\, \mathbf{s}_t, \boldsymbol{\theta}) = -\log \pi(a_t \,|\, \mathbf{s}_t, \mathbf{g}_\tau; \boldsymbol{\theta})(r_t + \gamma V_\theta(\mathbf{s}_{t+1}) - V_\theta(\mathbf{s}_t)) + \beta_H H_t(\pi) \quad (1)$$

$$\mathcal{J}_V(\mathbf{s}_t, \boldsymbol{\theta}) = \frac{1}{2}(V_\theta(\mathbf{s}_t) - R)^2 \quad (2)$$

and $R = \mathbb{E}_{\tau\sim\pi}[\sum_{i=t}^{T} \gamma^t r_i \,|\, \mathbf{s}_t]$. Here, H_t is the entropy of the policy that acts as a regularizer with β_H as its hyperparameter. The value function acts as a critic for the policy's generated actions. Typically random samples from the environment containing various room types are taken by rolling out the policy to obtain state-action-reward tuples to compute those terms. Note that the value function is the only information the agent has about the future potential of the current state during training. However, during the test time, the value function is not used by the agent anymore.

4. ForeSI

In this section we present details of our approach to enable explicit future sub-goal, or foresight, generation during

[1] Here, we drop the explicit dependency on the visual observations since our state $\mathbf{s}_t$ is learnt using a (recurrent) deep neural network that captures its representation.

the navigation.

4.1. Method Overview

Having a single policy with which to achieve various visual navigation tasks based solely on a target object's name is not trivial. The aim of our *Foresight Sub-goal Imaginator* (ForeSI) agent (see Figure 2) is to enable predicting (*i.e.*, imagining) a sub-goal state[2] through which the agent has maximal chances of success. Incorporating such a sub-goal in the policy enables the agent to take better actions since it has an indication of what states are important to visit in the future and eliminates a need for unnecessary exploration at test time. There are two main challenges to overcome: (1) to determine the sub-goal state to be learned

In determining the sub-goal state to be learned, we note that the sub-goal most valuably imagined is that with the highest impact towards receiving the maximum reward. In other words, the optimal sub-goal is the one with which we could learn the best value function to estimate the expected reward. Intuitively, when the agent reaches the optimal sub-goal, finding the target and receiving the maximum reward should be easy (see Section 4.2).

Once we find the optimal sub-goal, we can train a model to predict or *imagine* [3] an instance for an unseen environment. To that end, we collect a replay buffer of the *successful* trajectories to train our sub-goal generation module. Intuitively, when an agent navigates through particular states to achieve its goal, imagining the sub-goal from a related successful trajectory helps the agent to identify how to plan and take actions in unseen environments (see Section 4.3). This imagination is conditioned on the initial state and the target object and is integrated into a parallel execution of the policy in Asynchronous Advantage Actor-Critic (A3C) [23] for best performance. This allows the agent to learn to both imagine and navigate (see Section 4.4).

4.2. Hindsight: Learning to Identify Sub-Goal

In the first step, we consider learning to identify the sub-goal state through which a navigation episode is successful. To that end, the agent has to consider its current state and the sub-goals it has navigated through with their corresponding visual, target object and dynamics representation. Intuitively, for the agent to obtain a high reward in a straight path through a hallway, for instance, it should have chosen the best action while at the previous corner to be successful. Since the only aspect of the RL that considers the future reward is the value function, we modify Eq. (1) using a residual function of the past states as:

$$V_\theta(\mathbf{s}_t) \approx V_\theta(\mathbf{s}^\star_t), \qquad \mathbf{s}^\star_t = \sum_{j=0}^{t} \alpha_j v_{\boldsymbol{\omega}}(\mathbf{s}_j) + \mathbf{s}_t\,. \tag{3}$$

[2]Note that a sub-goal state could potentially be the final goal state.

[3]Note we use sub-goal generation, imagination and prediction interchangeably throughout this work.

Here, $v_{\boldsymbol{\omega}}(\mathbf{s}_j)$ is a linear function of the input and α_j is the jth dimension of $\boldsymbol{\alpha}$, which is defined as follows:

$$\boldsymbol{\alpha} = \text{softmax}\left(\frac{q_{\boldsymbol{\omega}}(\mathbf{s}_t) k_{\boldsymbol{\omega}}([\mathbf{s}_0 : \mathbf{s}_t])^\top}{\sqrt{t+1}}\right). \tag{4}$$

Here, $q_{\boldsymbol{\omega}}$ and $k_{\boldsymbol{\omega}}$ are linear functions analogous to the query and key in an attention mechanism [36] with $\mathbf{s}_{0:t}$ the concatenation of the states up to time t. We denote all of our sub-goal selection parameters by the set $\boldsymbol{\omega}$. Moreover, α_j is the correlation between state j and the current state t, and its magnitude specifies the likelihood that state j is an important sub-goal to reach.

Effectively, we use the attention mechanism described above to identify the sub-goal state that minimises the state value function estimation error. Using the key-query product above we choose the state in the past that is most related with the current state. Interestingly, if the state is novel (*i.e.*, uncorrelated with the past states) we allow the agent to assign a high attention weight to its most recent state. We select the state with the maximum attention weight as the sub-goal state at the end of an episode, *i.e.*, $\hat{\mathbf{s}}_\tau = \mathbf{s}_{t^*}$ where $t^* = \arg\max_t \alpha_t$ (note $\boldsymbol{\alpha}$ has length $T+1$). We subsequently task ForeSI to learn to generate this sub-goal. Using this method we ensure that the imagination will guide the policy towards a state that has the highest correlation to a successful goal state. Additionally, allowing for the sub-goal state to be different from the previously achieved goal state helps avoid repeating the potentially sub-optimal trajectories.

4.3. Foresight: Learning to Generate Sub-Goal

For learning to imagine, or generate, the selected sub-goal state $\hat{\mathbf{s}}_\tau$ we consider a replay buffer. The replay buffer, denoted by M is filled with tuples of $(\mathbf{s}_0, \mathbf{g}_\tau, \hat{\mathbf{s}}_\tau)$, an initial state, an embedding representation of the target object $\mathbf{g}_\tau$ for trajectory τ and the sub-goal. We then devise the following objective to train our sub-goal generation function $f_{\mathbf{w}}$:

$$\min_{\mathbf{w}} \quad \mathbb{E}_{(\mathbf{s}_0, \mathbf{g}_\tau, \hat{\mathbf{s}}_\tau)\sim M} \left| \hat{\mathbf{s}}_t - f_{\mathbf{w}}([\mathbf{s}_0 : \mathbf{g}_\tau]) \right| \tag{5}$$

where $\mathbf{w}$ is the set of parameters of our the sub-goal generation module and $[:]$ denotes concatenation of vectors. For $f_{\mathbf{w}}$ we use a multi-layer percepteron and a bottleneck with the intuition that the structure of the sub-goal distributions lies in a lower dimensional space. Intuitively, the imagination module does not need to generate every single dimension of the state accurately, since there might be unnecessary information about the other objects or the background scene in the representation. It might also include dynamic objects that constantly change location and/ or appearance across different environments. Using a smooth version of an L1 loss, we avoid penalising such inexact predictions too harshly.

We use a shared recurrent state encoding for both the policy and the sub-goal generation module. Therefore, the generated sub-goal state representation not only has information about the sub-goal's visual appearance but also encodes the history of the past observations and actions before that state. Hence, we essentially generate a representation of the whole trajectory that leads the agent to a successful goal state. Furthermore, sharing the state encoder that is trained along with the policy helps generate sub-goal states that are directly useful for the action selection. This is as compared to methods such as [10] where the agent needs to first perform a random policy search to collect a dataset from the environment, and then use only the visual features for generating future states.

Since computing the expectation in Eq. (5) is impractical due to the buffer size and the constant change in the distribution of states, we only consider the latest collected tuples to update the model's parameters. That is, we update the model when $|M| = M_{\max}$ and empty the buffer.

4.4. RL Integration

To use ForeSI we condition the policy on the imagined sub-goal state. This way the policy learns to take actions based on the current state and the imagined one to achieve its goal. One potential issue with this approach is that it could bias the policy towards exploiting the known imagined sub-goal states and avoiding essential exploration, hence degrading the performance. We address this issue by adding Gaussian noise to the imagined states. Therefore, in practice the generated sub-goal is as follows:

$$\begin{aligned} &a_t \sim \pi_\theta(a_t \,|\, \mathbf{s}_t, \mathbf{g}_\tau, \mathbf{I}_\tau) \quad \text{and,} \\ &\mathbf{I}_\tau = f_{\mathbf{w}}([\mathrm{s}_0 : \mathbf{g}_\tau]) + \boldsymbol{\eta}, \quad \boldsymbol{\eta} \sim \mathcal{N}(\mathbf{0}, \sigma^2 \boldsymbol{I}) \end{aligned} \tag{6}$$

Here, $\mathbf{0}$ is a vector of all zeros, $\boldsymbol{I}$ is the identity matrix, and σ^2 is the variance of the noise. The additive noise is decayed during training to allow more exploration when the agent is more unsuccessful (*i.e.*, the imagination is not robust enough) and exploit otherwise. We choose the noise level by adjusting the variance, that is,

$$\sigma^2 = \max(\sigma^2_{\max} - \rho, 0) \tag{7}$$

where $\sigma^2_{\max}$ is a pre-defined maximum variance threshold, and ρ is a moving average of the success rate over the past episodes. This simple heuristic ensures the noise level is proportionate to the success rate, for instance, if $\rho = 0.9$ and success rate is around 90% we completely remove the added noise.

Finally, we integrate the sub-goal selection and the imagination into the on-policy Asynchronous Advantage Actor-Critic (A3C) [23] algorithm which allows for efficient and parallel training of multiple agents. We revise the training objectives to integrates ForeSI into A3C:

$$\begin{aligned} \mathcal{J}^\star_\pi(a_t \,|\, \mathbf{s}_t, \boldsymbol{\theta}) = &- \log \pi(a_t \,|\, \mathbf{s}_t, \mathbf{g}_\tau, \mathbf{I}_\tau; \boldsymbol{\theta})(r_t + \\ &\gamma V_\theta(\mathbf{s}^\star_{t+1}) - V_\theta(\mathbf{s}^\star_t)) + \beta_H H_t(\pi) \end{aligned} \tag{8}$$

$$\mathcal{J}^\star_V(\mathbf{s}_t, \boldsymbol{\theta}) = \frac{1}{2}(V_\theta(\mathbf{s}^\star_t) - R)^2 \tag{9}$$

We summarise the training of our approach in Algorithm 1. As may be observed, in the spirit of A3C, each agent stores its own replay buffer and computes its noise level hyperparameter proportionate to its moving average success rate.

Algorithm 1: Training One ForeSI Agent

```
Randomly initialise θ, w, ω
Initialise replay buffer M = ∅
σ² = σ²_max                                   ▷ Imagination Noise
while episode < MAX_EPISODE do
    (s_0, g_τ) ~ E_RND      ▷ E_RND is a random environment
    I_τ = f_w([s_0 : g_τ]) + η,  η ~ N(0, σ² I)      ▷ Eq. (6)
    while a_t ≠ STOP & t ≤ T do
        a_t ~ π_θ(a | s_t, g_τ, I_τ)
    end
    Compute α using eq. (4) for the trajectory
    Update θ and ω via eq. (8) and (9)
    if trajectory is successful in the environment then
        ŝ_τ = s_{t*},  t* = arg max_t α_t
        M = M ∪ {s_0, g_τ, ŝ_τ}                  ▷ Update Buffer
    Update the average success rate ρ
    if |M| = M_max then
        for epoch ≤ Epoch_max do
            for {s_0, g_τ, ŝ_τ} ∈ M do
                ω ← ω − β∇_w |ŝ_t − f_w([s_0 : g_τ])|
            end
        end
        M = ∅
    Update σ²                                     ▷ Eq. (7)
end
```

5. Experiments

In this section we present implementation details of our method as well as extensive experiments to show how ForeSI improves the visual navigation performance of multiple significantly different baselines.

5.1. Experimental Setup

We use the AI2THOR [20] environment to benchmark our results. The simulator consists of photo-realistic indoor environments (*i.e.*, houses) categorised into four different room types: kitchen, bedroom, bathroom and living room. We run our experiments on two distinct setups of this simulator for fair comparison against previous state-of-the-art approaches. In both of those setups 20 different scene layouts of each room type are used for training; 5 scenes for validation and 5 for the test. We provide the final results on the test set based on the best performing model on the

Method	SPL	SR	SPL>5	SR>5
First Setup				
A3C [39]	14.68	33.04	11.69	21.44
A3C+MAML [39]	16.15 $\pm$0.5	40.86 $\pm$1.2	13.91 $\pm$0.5	28.70 $\pm$1.5
A3C+ForeSI	**15.23** $\pm$0.4	**36.80** $\pm$1.1	**13.14** $\pm$0.3	**27.55** $\pm$1.4
A3C+MAML+ForeSI	**16.75** $\pm$0.5	**45.5** $\pm$1.0	**15.8** $\pm$0.6	**34.7** $\pm$1.1
Second Setup				
A3C+ORG [7]	37.5	65.3	36.1	54.8
A3C+ORG+ForeSI	**39.41** $\pm$0.3	**68.0** $\pm$0.6	**36.85** $\pm$0.4	**56.11** $\pm$0.8

Table 1. A quantitative comparison of our method against four representative baselines. The baseline methods are significantly different in perception, policy modelling and environment setup. ForeSI improves the performance of all of them including the previous state-of-the-arts [7, 39]. *SPL>5* and *SR>5* show the metrics for trajectories longer than 5 steps.

validation set. For a fair comparison, we follow the exact object configuration and target object list according to the baseline methods.

In the first setup, we follow the configuration as used in [39,41]. In this setup the target object is selected from the following list: pillow, laptop, television, garbage can, box, bowl, toaster, microwave, fridge, coffee maker, garbage can, plant, lamp, book, alarm clock, sink, toilet paper, soap bottle and light switch. Here a trajectory is considered successful if the agent stops within **1 meter** circular proximity of the target object while the object is visible in the egocentric view of the agent.

In the second setup, we follow the configuration recently introduced in [7]. The distribution of the target objects and the object location configuration in this setup is significantly different. The following objects are added to the list of the targets compared to the first setup: cellphone, chair, desk lamp, floor lamp, kettle, pan, plate, pot, remote control, stove burner; moreover, the following objects are removed from the list in the first setup: pillow, box, plant, lamp, toilet paper, soap bottle. The difference in the target objects can also affect the navigation performance due to differences in size, distant visibility etc. Lastly, here the successful trajectory criterion is relaxed to a **1.5 meters** circular distance around the target object.

Following the recent conventions in visual navigation tasks [1, 7, 39, 41] we evaluate the performance of our method using two main metrics: Success Rate (SR) and Success weighted by inverse Path Length (SPL), calculated as $\sum s_i \frac{L_i^g}{L_i}$, where L_i^g is the ground truth shortest path to the target, L_i is the length of the trajectory as taken by the agent and s_i is the binary success indicator.

5.2. Implementation Details

We use A3C [23] as the basis of our method. This actor-critic algorithm is a good fit for visual navigation tasks since the agents can explore in parallel and asynchronously, which is more computationally efficient. Our method can potentially be integrated into other actor-critic algorithms since it only relies on the critic to identify the sub-goals during the training.

Our backbone model comprises a single layer LSTM state encoder with 512 hidden states. The input to the encoder at each time step is the visual features extracted from a pre-trained ResNet-18 [14] and the Glove embedding [15] of the target object, as visualised in Figure 2. Note the Glove embedding in the second setup is replaced with a one-hot vector embedding following the settings in [7]. The policy comprises of a single fully-connected layer that outputs the distribution over 6 actions, {*RotateLeft, RotateRight, MoveAhead, LookDown, LookUp, Stop*}, using a Softmax activation function. The state-value head is a two-layer MLP that maps the attended state representation to a single scalar value. We use a reward of 5 for task completion and a negative step penalty of -0.01 for each taken action. We implement the attention mechanism as three fully connected layers of size 512 that map the query, key and value.

Our ForeSI's $f_{\mathbf{w}}$ comprises 6 fully connected layers that receive the concatenation of the state representation and the target object embedding as input. We use a $\tanh$ nonlinearity in all layers to mimic the behaviour of the LSTM state encoder.

We use a replay buffer of size 32 for each agent to train its sub-goal generation module separately. We, then, transfer the learnt weights to a shared model between the agents, asynchronously, and we empty the replay buffer. For the additive noise in sub-goal generation (see Section 4.4), we use $\sigma^2_{max} = 0.9$ and the last 100 episodes to compute the moving average for the success rate.

5.3. Results

In Table 1 and Table 2 we compare our approach to that of state-of-the-art baselines in two separate simulator setups as discussed in Section 5.1. The first baseline, **A3C**, is only using the backbone model described in Section 5.2 trained using A3C RL algorithm. This is the main baseline that shows how a simple RL objective can perform without any extra components or modifications. **A3C+ForeSI** is the variant of A3C upgraded with our method. We see that em-

Method	Bathroom	Bedroom	Kitchen	Living
First Setup				
+MAML	**28.49**/69.6	8.65/**29.2**	17.8/43.6	7.71/21.6
+MAML+ForeSI	27.03/**73.6**	**8.81**/27.6	**21.55/54.0**	**9.61/26.8**
Second Setup				
+ORG	**49.87/83.89**	35.43/62.21	38.63/69.02	29.33/47.83
+ORG+ForeSI	47.44/80.4	**38.15/65.6**	**41.09/72.87**	**30.39/52.14**

Table 2. Detailed comparison against previous state-of-the-art methods on the two different setups; SPL/SR are reported per room type. Our method is general enough to improve performance in 3/4 of the room types, with a marginal performance impact on 1/4. Notably, we improve the SR of trajectories in the "kitchen" and "living room" by a large margin where the trajectories are generally longer and generating the sub-goal is more important.

powering the agent with forward modelling improves the success rate by more than 3%. The improvement on longer-horizon tasks, *i.e.*, longer trajectories, is even more significant, more than 6%. We conjecture that this is mainly because in longer trajectories, in the absence of ForeSI, the agent can more easily forget the objective by focusing more on short term tasks such as obstacle avoidance. Furthermore, in longer trajectories, it is, also, more likely to follow a path that does not lead to success from the very beginning when ForeSI is not used. Using ForeSI, however, helps the agent to constantly remember the (sub-)goal and, thus, take the actions that are more likely to achieve the goal.

Additionally, we, also, evaluate the performance of our approach when used along with a self-adaptive baseline, where we utilise meta-learning similar to **A3C+MAML** [39] for test-time policy adaptation. **A3C+MAML** shows a considerable performance boost over the simple **A3C** baseline as shown in Table 1. Despite that, when combined with our method in **A3C+MAML+ForeSI** we observe an additional absolute improvement of around 5% in success rate on both the short and long trajectories. This further demonstrates the modularity of our approach by showing that it improves both adaptive, and non-adaptive, baseline methods. Note that while the absolute improvement in both the short and long trajectories present similar figures, the relative improvement is much larger for long trajectories. This, again, supports the previous hypothesis that our imagination can effectively help address both the LSTM forgetting problem and finding the correct path to the successful target. Furthermore, nearly 2% improvement compared to **A3C+MAML+ForeSI** on the SPL over the long trajectories implies that ForeSI helps avoid futile wandering around.

The third baseline that we compare our method against is **A3C+ORG** [7]. In that method the authors use an off-the-shelf object detector, *i.e.* FasterRCNN [30], and incorporate the detected object bounding boxes along with their confidence scores into the policy by building a neural graph, inspired by [41]. Their method builds a better state representation that renders finding target objects much easier for the agent. As long as the objects are detected by the object detector the agent can learn to directly navigate towards them taking a relatively short trajectory. The authors also modify the environment setup as discussed in Section 5.1. Therefore, we find that a suitable baseline to compare against to demonstrate the modularity of our approach. We build our approach using their method as the backbone which we call **A3C+ORG+ForeSI**. We observe that our method improves the success rate by over %2 compared to [7]. This shows that irrespective of the quality of the state representations empowering the agent with a forward model using our method can help improve the navigation performance. Note that we do not further compare our results with the addition of imitation learning into this problem, as the authors did in [7]. This is because, in this paper, we are only concerned about integrating a forward model into model-free RL.

5.4. Ablation Studies

What to Imagine: In our approach, we learn to generate the sub-goal state from the agent's own successful navigation episode; we could, however, consider alternative approaches that we compare here. Firstly, we seek to answer whether our ForeSI provides the agent with valuable information about its future sub-goal states.

In Table 3, we consider a random state generation by replacing the output of our sub-goal generation with $\tanh(\mathbf{s})$ where $\mathbf{s}$ is sampled from a Gaussian noise, *i.e.* $\mathbf{s} \sim \mathcal{N}(\mathbf{0}, 0.5\mathbf{I})$, shown in **Ours-RND**. The $\tanh$ non-linearity assures that the noise is similar in values to the states generated by our sub-goal generation, but are nonetheless rather random and meaningless. As observed, this leads to a deterioration in the success rate compared to the original baseline, **A3C+ORG**. We hypothesise that, because it is uninformative, the policy learns to dismiss the randomly generated state to some extent, but not completely, hence the slight performance degradation.

We further compare our approach with the case of predicting the weighted average of the states in Equation 3, rather than a single sub-goal. The intuition for this experiment is that knowing the attended states gives the agent valuable information about what important future states are expected to be traversed. As shown in **Ours-ATT** of Table 3 this leads to improved performance which indicates it is helpful to know about the sub-goal states ahead of the agent; however, since the sub-goal is not always identifiable

Method	SPL	SR
A3C+ORG [7]	37.5	65.3
Ours-RND	37.57	64.8
Ours-INT	37.78	63.8
Ours-ATT	37.76	65.4
Ours-ForeSI	**38.66**	**67.6**

Table 3. Ablation study of different variants of our method. ForeSI avoids the unnecessary complications of multi-step future predictions. Our future state predictions are meaningful.

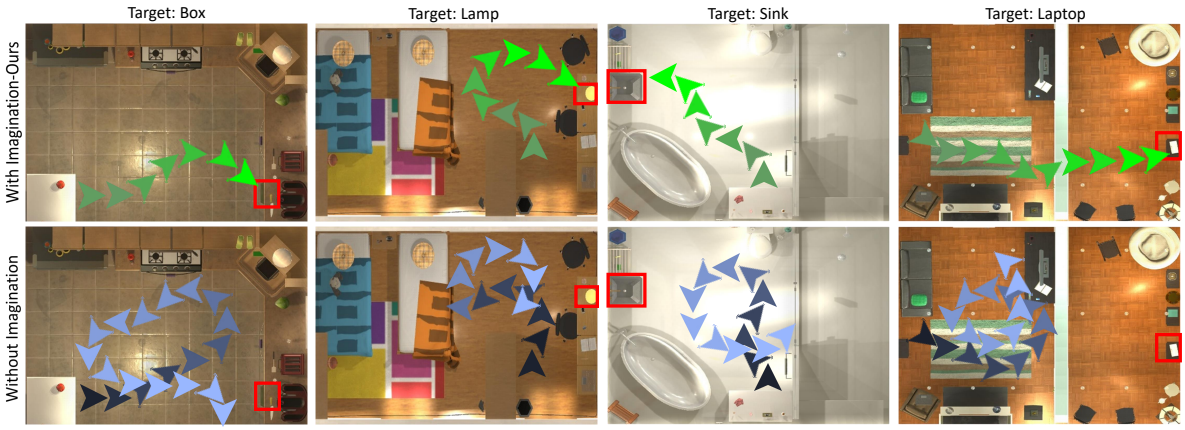

Figure 3. Qualitative comparison of navigation episodes of ForeSI against the baseline method [39] in four different room types. The examples represent cases where our method improves both the robustness and the efficiency of navigation. For more examples and failure refer to supplementary material.

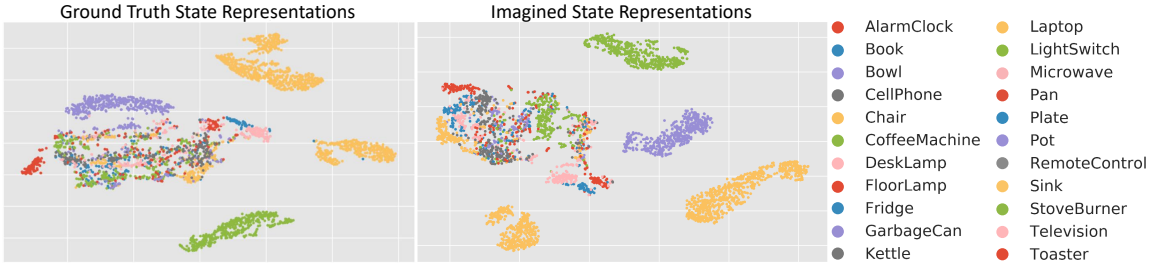

Figure 4. t-SNE comparison of the sub-goal states generated using our sub-goal method against the ground truth states. Our agent's predicted sub-goals closely follow the structure of the ground-truth and are clustered similarly.

(multiple sub-goals and trajectories could have the same attended states) the performance gain is rather insignificant.

Sub-Goal Generation Interval: Rather than generating the sub-goal from the initial state, we can consider generating multiple sub-goals in fixed intervals every given number of steps. In that case, in each interval step, a different sub-goal is predicted. As shown in **Ours-INT** in Table 3, this leads to a performance degradation. We believe this is because the policy collapses to a deterministic one due to the constraints on predicting multiple sub-goals. Consequently, the agent stops exploring and converges to a sub-optimal solution.

Explicitly Structured Sub-Goal Generation: It can be seen in Figure 4 that ForeSI can accurately learn the structure in the state representations used by the agent's state encoder. To further investigate the quality of the generated states we train a Conditional Variational Auto-Encoder (C-VAE) [34] to learn to encode the state representations. Then, we use ForeSI to generate the latent state of the C-VAE rather than working directly on the agent's state representations. We compare the average loss achieved by our module trained directly on the agent's state representations with that of trained on the latent representations of the C-VAE. We observe that direct sub-goal generation achieves a lower average loss, 0.012 vs 0.018, while being significantly less complex.

5.5. Qualitative Comparison

In Figure 3 we present four sample navigation episodes where our method improves both the efficiency and robustness of the baseline [39]. We refer the reader to Table 1 for the overall test set results; more navigation examples, including the failure cases, can be found in the supplementary material.

6. Conclusion and Future Work

In this work, we showed that integrating a forward model into visual navigation using ForeSI improves both the robustness and efficiency of navigation. We achieved this by enabling our agent to imagine (*i.e.* to predict) a future sub-goal state that leads to a successful navigation episode. Our extensive experiments show how ForeSI can be integrated into a wide variety of methods to improve their navigation performance. While it has been proven effective, we consider this work as an initial step towards enabling forward modelling for visual navigation. One interesting future direction may be to investigate the role of sub-goal generation for better exploration. Moreover, we may also improve the sub-goal generation by better learning from failed episodes due to inaccurate sub-goals.

References

[1] Peter Anderson, Qi Wu, Damien Teney, Jake Bruce, Mark Johnson, Niko Sünderhauf, Ian Reid, Stephen Gould, and Anton van den Hengel. Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3674–3683, 2018.

[2] Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, OpenAI Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. *Advances in neural information processing systems*, 30:5048–5058, 2017.

[3] Matthew Chang, Arjun Gupta, and Saurabh Gupta. Semantic visual navigation by watching youtube videos. *arXiv preprint arXiv:2006.10034*, 2020.

[4] Devendra Singh Chaplot, Dhiraj Gandhi, Saurabh Gupta, Abhinav Gupta, and Ruslan Salakhutdinov. Learning to explore using active neural slam. *arXiv preprint arXiv:2004.05155*, 2020.

[5] Devendra Singh Chaplot, Dhiraj Prakashchand Gandhi, Abhinav Gupta, and Russ R Salakhutdinov. Object goal navigation using goal-oriented semantic exploration. *Advances in Neural Information Processing Systems*, 33, 2020.

[6] Matt Deitke, Winson Han, Alvaro Herrasti, Aniruddha Kembhavi, Eric Kolve, Roozbeh Mottaghi, Jordi Salvador, Dustin Schwenk, Eli VanderBilt, Matthew Wallingford, et al. Robothor: An open simulation-to-real embodied ai platform. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3164–3174, 2020.

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

[8] Heming Du, Xin Yu, and Liang Zheng. Vtnet: Visual transformer network for object goal navigation. *arXiv preprint arXiv:2105.09447*, 2021.

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

[10] David Ha and Jürgen Schmidhuber. World models. *arXiv preprint arXiv:1803.10122*, 2018.

[11] Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination, 2020.

[12] Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. In *International Conference on Machine Learning*, pages 2555–2565. PMLR, 2019.

[13] Danijar Hafner, Timothy Lillicrap, Mohammad Norouzi, and Jimmy Ba. Mastering atari with discrete world models. *arXiv preprint arXiv:2010.02193*, 2020.

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

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

[16] Lukasz Kaiser, Mohammad Babaeizadeh, Piotr Milos, Blazej Osinski, Roy H Campbell, Konrad Czechowski, Dumitru Erhan, Chelsea Finn, Piotr Kozakowski, Sergey Levine, et al. Model-based reinforcement learning for atari. *arXiv preprint arXiv:1903.00374*, 2019.

[17] Lukasz Kaiser, Mohammad Babaeizadeh, Piotr Milos, Blazej Osinski, Roy H Campbell, Konrad Czechowski, Dumitru Erhan, Chelsea Finn, Piotr Kozakowski, Sergey Levine, et al. Model-based reinforcement learning for atari. *arXiv preprint arXiv:1903.00374*, 2019.

[18] John Kanu, Eadom Dessalene, Xiaomin Lin, Cornelia Fermuller, and Yiannis Aloimonos. Following instructions by imagining and reaching visual goals, 2020.

[19] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[20] 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.

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

[22] 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.

[23] 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.

[24] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[25] Mahdi Kazemi Moghaddam, Qi Wu, Ehsan Abbasnejad, and Javen Shi. Optimistic agent: Accurate graph-based value estimation for more successful visual navigation. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 3733–3742, 2021.

[26] Ashvin V Nair, Vitchyr Pong, Murtaza Dalal, Shikhar Bahl, Steven Lin, and Sergey Levine. Visual reinforcement learning with imagined goals. In *Advances in Neural Information Processing Systems*, pages 9191–9200, 2018.

[27] Suraj Nair and Chelsea Finn. Hierarchical foresight: Self-supervised learning of long-horizon tasks via visual subgoal generation. *arXiv preprint arXiv:1909.05829*, 2019.

[28] Soroush Nasiriany, Vitchyr Pong, Steven Lin, and Sergey Levine. Planning with goal-conditioned policies. In *Advances in Neural Information Processing Systems*, pages 14843–14854, 2019.

[29] Aske Plaat, Walter Kosters, and Mike Preuss. Model-based deep reinforcement learning for high-dimensional problems, a survey. *arXiv preprint arXiv:2008.05598*, 2020.

[30] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

[31] Zhizhou Ren, Kefan Dong, Yuan Zhou, Qiang Liu, and Jian Peng. Exploration via hindsight goal generation. In *Advances in Neural Information Processing Systems*, pages 13485–13496, 2019.

[32] 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.

[33] Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, et al. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588(7839):604–609, 2020.

[34] Kihyuk Sohn, Honglak Lee, and Xinchen Yan. Learning structured output representation using deep conditional generative models. In *Advances in neural information processing systems*, pages 3483–3491, 2015.

[35] Yunhao Tang and Alp Kucukelbir. Hindsight expectation maximization for goal-conditioned reinforcement learning, 2020.

[36] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[37] Théophane Weber, Sébastien Racanière, David P. Reichert, Lars Buesing, Arthur Guez, Danilo Jimenez Rezende, Adria Puigdomènech Badia, Oriol Vinyals, Nicolas Heess, Yujia Li, Razvan Pascanu, Peter Battaglia, Demis Hassabis, David Silver, and Daan Wierstra. Imagination-augmented agents for deep reinforcement learning, 2018.

[38] Luca Weihs, Jordi Salvador, Klemen Kotar, Unnat Jain, Kuo-Hao Zeng, Roozbeh Mottaghi, and Aniruddha Kembhavi. Allenact: A framework for embodied ai research. *arXiv preprint arXiv:2008.12760*, 2020.

[39] 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.

[40] 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.

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

[42] Lunjun Zhang, Ge Yang, and Bradly C Stadie. World model as a graph: Learning latent landmarks for planning. *arXiv preprint arXiv:2011.12491*, 2020.

[43] Marvin Zhang, Sharad Vikram, Laura Smith, Pieter Abbeel, Matthew Johnson, and Sergey Levine. Solar: Deep structured representations for model-based reinforcement learning. In *International Conference on Machine Learning*, pages 7444–7453. PMLR, 2019.

[44] 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.