

# Learning for Visual Navigation by Imagining Success

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

Visual navigation is often cast as a reinforcement learning (RL) problem. Current methods typically result in a sub-optimal policy that learns general obstacle avoidance and search behaviours. For example, in the target-object navigation setting, the policies learnt by traditional methods often fail to complete the task, even when the target is clearly within reach from a human perspective. In order to address this issue we propose to learn to imagine a latent representation of the successful (sub-)goal state. To do so, we have developed a module which we call Foresight Imagination (ForeSIT). ForeSIT is trained to imagine the recurrent latent representation of a future state that leads to success, e.g. either a sub-goal state that is important to reach before the target, or the goal state itself. By conditioning the policy on the generated imagination during training, our agent learns how to use this imagination to achieve its goal robustly. Our agent is able to imagine what the (sub-)goal state may look like (in the latent space) and can learn to navigate towards that state. We develop an efficient learning algorithm to train ForeSIT in an on-policy manner and integrate it into our RL objective. The integration is not trivial due to the constantly evolving state representation shared between both the imagination and the policy. We, empirically, observe that our method outperforms the state-of-the-art methods by a large margin in the commonly accepted benchmark AI2THOR environment. Our method can be readily integrated or added to other model-free RL navigation frameworks.

## 1. Introduction

The ability for an intelligent agent to navigate through an environment to carry out human provided instructions is one of the primary objectives of robotics and artificial intelligence. Visual navigation has emerged to enable such agents to learn (e.g. using deep learning) to move towards a given target object [6]. The problem is then defined as a deep reinforcement learning [36] problem since the agent needs to

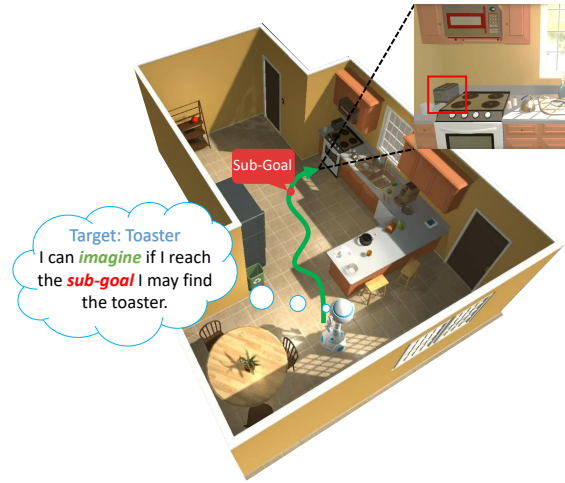


Figure 1. Enabling an agent to imagine states on the path to success improves its ability to carry out complex tasks, particularly in unseen environments. As opposed to the conventional approaches, our ForeSIT takes actions not only based on the current state, but also the potential future, to achieve its goal.

learn a navigation policy and stopping criteria conditioned on finding a specific target. It is a particularly challenging problem since the agent needs to learn how to avoid obstacles and take actions by distinguishing similar but visually different targets in a complex environment. Furthermore, there are typically multiple action sequences (i.e. trajectories) that could lead to a successful navigation at each starting state, let alone the trajectories that might fail. Learning to select the right action at each time step to create a trajectory that leads to the goal is the primary challenge.

Furthermore, in navigation the agent only receives a delayed reward, after task completion, hence knows the reasonableness of its actions in *hindsight*. Explicitly incorporating the transition in the environment for better prediction of the potential outcome of the actions, hailed model-based RL, is also developed. However, model-based approaches are generally data hungry and harder to train since every state transition in the environment has to be accurately modelled. In addition, it is impossible for the agent to identify

which specific actions had the greatest impact on its success; for navigation, for instance, it does not know whether avoiding an obstacle, going straight in a hallway, or stopping near a book was the most important decision. This lies at the heart of RL which seeks to solve the problem of credit assignment from delayed environmental feedback.

To mitigate the above mentioned issues we propose our *Foresight Sub-goal Imagination unit* (ForeSIT). Intuitively, as shown in Figure 1, our imagination module generates a representation of a potential successful trajectory given the target and the initial state (*i.e.* the egocentric view of the environment). ForeSIT provides the agent with foresight of the future sub-goal through which the agent could successfully achieve its target. In a way our approach alleviates the problem with model-free RL in visual navigation by allowing the agent to incorporate a foresight credit assignment. This represents a significant advantage.

Our imagination module helps the agent in multiple ways: first, it provides the agent with an imagined representation of the sub-goal state that will help with successful task completion. Second, it helps the agent exploit its memory of successful traversed states, more efficiently. Thirdly, it helps the agent to perform credit assignment better than conventional heuristic based methods by helping the agent better learn the distribution of the states that lead to higher final reward.

We train our ForeSIT using the successful trajectories collected by the agent and integrate into a navigation policy, in particular an on-policy actor critic algorithm [17]. We learn the sub-goals as states that highly correlate with successful navigation using an attention mechanism. We then train a model to predict the highest attended state representation, signifying its impact on success. That way, even if that single collected trajectory is sub-optimal, our ForeSIT module could potentially generate a more optimal sub-goal by learning from other more successful ones. Moreover since our ForeSIT is akin to a plugin for RL in a navigating agent, the imagination module and the main policy evolve collaboratively. This is particularly desirable for smooth integration into different RL approaches without extra environments, or algorithm specific requirements.

We gradually introduce imagination into the policy so that the agent performs sufficient exploration, as well as learning to imagine accurately. Furthermore, we modify the value function to operate on a representation of the trajectory rather than an individual state using an attention-based internal memory. Practically, we share the state space between our imagination module, and the policy we define, using a deterministic Long Short Term Memory (LSTM). The LSTM encodes a history of the observation and actions in each trajectory. Therefore, imagining sub-goal states in the state representation of the policy is effectively imagining a trajectory leading to success. We find that our simple

ForeSIT module works well on multiple baseline and state-of-the-art models in the AI2THOR environment [13].

Our primary contributions are as follows:

1. We propose an imagination module ForeSIT that is able to generate foresight of states on the path to success, conditioned on the first state and the target object.
2. We propose a method to efficiently integrate the ForeSIT module into reinforcement learning which does not require pre-exploration for data collection.
3. We show the effectiveness of our method in improving visual navigation performance when added to multiple different baselines, including previous state-of-the-art methods.

## 2. Related Work

**Visual Navigation** Recently, visual navigation research has gained momentum due to the availability of high quality simulation environments [1, 13, 25, 34]. Different tasks [4, 15, 21, 22, 28, 30, 32] and ideas [14, 33, 35, 36] have been explored. Some recent methods use imitation learning [1, 4, 30] in simulation environments that provide ground truth state-action pairs. A more promising area of research, however, is to apply RL in the absence of ground truth annotations [33, 35, 36]. Our method introduces imagination for RL-based visual navigation for the first time. Therefore, we compare our method with related works in other tasks and environments.

**Experience Replay** Modifying the distribution of trajectories in a replay buffer, known as hindsight experience replay, has been explored in various works [3, 24, 27]. In this body of work the core idea is to modify the distribution of goal states in the replay buffer (*e.g.* hindsight) to train a more generalisable policy able to reach arbitrary goals provided during test time. In contrast, here we introduce a method that is trained in hindsight and tested in foresight to generate successful (sub-)goal states for navigation.

**Goal-Conditioned RL** Our method is also closely related to the area of goal-conditioned RL where the policy is conditioned on a given goal state. In [20] the authors use combinatorial optimisation on the states encoded using a Variational Auto-Encoder [12] 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 [18], an off-policy RL algorithm. In [19] the authors use self-imagined goals to train the agent to reach to arbitrary goals. Most recently, in [11] the authors use a VAE to learn to generate goals during test time. While these ideas are related to that proposed here, there are important differences: in contrast to [11] we train our method along with an on-policy RL algorithm; the constantly evolving shared state representation renders the task very challenging. In [11] the training and testing environment are simple and similar in distribution, limited to

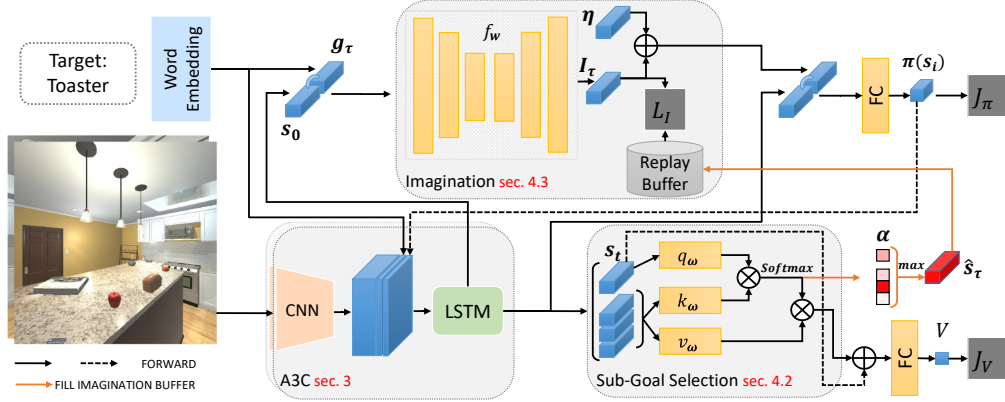


Figure 2. Our ForeSIT module efficiently augments actor-critic RL and is trained along with the policy sharing the state representation. We use a dot-product attention mechanism to identify the sub-goal state that leads to maximum reward and train our ForeSIT module to generate that state.

moving pucks. Finally, in their method only the final state is generated as the target while we use an attention mechanism to identify the optimal sub-goal state.

**Model-Based RL** Our ForeSIT module generates future state representations, and thus is related to model-based RL. In [31] the authors train a recurrent model of the environment dynamics that can be used for model-based planning by unrolling a trajectory in the imagination. In a similar but more recent method [8] the authors train the policy purely on imagined trajectories. In all of these methods the environment model is a recurrent reconstruction-based model inspired by [7]. These state encoders are not directly trained for policy optimisation and thus can be sub-optimal. Moreover, the mentioned works have only been tested on toy environments and the generalisation to real 3D tasks is a big question. In contrast, we enable imagination for more challenging on-policy RL in complex navigation environments.

### 3. Background

In visual navigation the objective for an agent starting from a random initial location is to take a sequence of movement actions (collectively, a trajectory) to reach a named target object using only observed ego-centric RGB visual input from the environment. An episode of this navigation trajectory ends if either the agent takes the stop action or the maximum number of permitted actions is achieved. A successful trajectory sees the agent stop 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  $\{O, S, A, \mathcal{G}, \mathcal{P}, r, \gamma\}$  [16, 33]. 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,  $\mathcal{P} := p(s_t | s_{t-1}, a_{t-1})$  is the transition function or environment dynamics model (unknown) for state  $s_t \in S$ ,

and  $r$  is the reward function and  $\gamma$  is the reward discount factor.

Formally, a trajectory  $\tau$  of length  $T + 1$  consists of a tuple  $(s_0, a_0, r_0; s_1, a_1, r_1; \dots, s_T, a_T, r_T)$  that is generated by taking action  $a_t$  at time  $t$  and observing the next state according to the dynamics of the environment  $s_{t+1} \sim p(s_{t+1} | s_t, a_t)$ . A reward  $r_t = r(s_t, a_t, s_{t+1})$  is received from the environment at each time step, which is conventionally [33, 36] defined as a large positive number for a successful trajectory and a negative penalty otherwise.

A common approach is to use actor-critic RL methods for learning the optimal policy<sup>1</sup> [33, 35, 36] for navigation  $\pi_\theta(a_t | s_t, g_\tau, \theta)$  to choose action  $a_t$  at time  $t$  conditioned on an embedding of the target object’s name  $g_\tau$  (e.g. using GloVe embedding [10]) for a given environment (*i.e.* a room in AI2Thor)  $E$ . We use  $\theta$  to denote the set of all the parameters for RL. Learning involves minimising  $\mathcal{J}_\pi(a_t | s_t, \theta)$  the negative of expected advantage function while minimising the difference of the estimated value function and the true return  $\mathcal{J}_V(s_t, \theta)$  where we have:

$$\mathcal{J}_\pi(a_t | s_t, \theta) = -\log \pi(a_t | s_t, g_\tau; \theta)(r_t + \gamma V_\theta(s_{t+1}) - V_\theta(s_t)) + \beta_H H_t(\pi) \quad (1)$$

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

and  $R = \mathbb{E}_{\tau \sim \pi}[\sum_{i=t}^T \gamma^i r_i | s_t]$ . Here,  $H_t$  is the entropy of the policy that acts as a regularizer with  $\beta_H$  as its hyperparameter. The value function acts as a critic for the policy’s generated actions. Typically random samples from the environments containing various room types are taken by rolling out the policy to obtain state-action-reward tuples to compute these terms. Note that the value function is the only information the agent has about the future potential of

<sup>1</sup>Here, we drop the explicit dependency on the visual observations since our state  $s_t$  is learnt using a (recurrent) deep neural network that captures its representation.

the current state during training. However, during the test time, the value function is not used by the agent any more.

## 4. Foresight Sub-goal Imagination

In this section we detail our approach to providing RL with explicit foresight, or imagination, for navigation.

### 4.1. Imagination Module Overview

Having a single policy with which to achieve various visual navigation tasks based solely on a target object’s name is non-trivial. The aim of our *Foresight Sub-goal Imagination* (ForeSIT) module (see Figure 2) is to provide the agent with an imagined sub-goal state<sup>2</sup> such that it is a future state through which the navigation has maximal chance of success. Incorporating such a sub-goal in the policy enables the agent to take better actions since it has an indication of what likely states to visit and eliminates a need for unnecessary exploration at test time. There are two main challenges to overcome: (1) determining the sub-goal state to be learned, and (2) imagining or generating that sub-goal given the initial state and target object in an unseen environment.

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* an instance for an unseen environment. To that end, we collect a replay buffer of the successful trajectories to train our ForeSIT. Intuitively, when an agent navigates through particular states to achieve its target, imagination of 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, and is integrated into a parallel execution of the policy in Asynchronous Advantage Actor-Critic (A3C) [17] for best performance. This allows the agent to learn to both imagine and navigate (see Section 4.4).

### 4.2. Sub-goal Selection

In the first step we consider learning to identify the sub-goal state through which a navigation trajectory 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 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

<sup>2</sup>Note that a sub-goal state could potentially be the final goal state.

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:

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

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

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

Here,  $q_\omega$  and  $k_\omega$  are linear functions analogous to the query and key in an attention mechanism [29] 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  $\omega$ . Moreover,  $\alpha_j$  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 a trajectory, *i.e.*  $\hat{\mathbf{s}}_\tau = \mathbf{s}_{t^*}$  where  $t^* = \arg \max_t \alpha_t$  (note  $\alpha$  has length  $T+1$ ). We subsequently task ForeSIT 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. It is also easy to integrate into the existing RL approach.

### 4.3. Learning to Imagine

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  $\tau$  and the sub-goal. We then devise the following objective to train an imagination function  $f_\mathbf{w}$ :

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

where  $\mathbf{w}$  is the set of parameters of our ForeSIT and  $[\cdot]$  denotes concatenation of vectors. For  $f_\mathbf{w}$  we use a multi-layer perceptron 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. 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 ForeSIT module. Therefore, the state representation generated by the ForeSIT module not only has the information about the sub-goal state’s visual observation, but also encodes the history of the past observation and action pairs before that state. Hence, ForeSIT essentially generates 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 goal-states that are directly useful for the action selection. This is as compared to methods such as [7] where the agent needs to first perform random policy search to collect images 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. Imagination for Actor-Critic models

To use ForeSIT 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 the policy could bias towards only exploiting the known imagined sub-goal states and avoid exploration. This could lead to the agent’s policy collapsing to a deterministic one which is highly undesirable. We address this issue by adding Gaussian noise to the imagined states. Hence we have the revised policy:

$$a_t \sim \pi_\theta(a_t | \mathbf{s}_t, \mathbf{g}_\tau, \mathbf{I}_\tau) \quad \text{and},$$

$$\mathbf{I}_\tau = f_{\mathbf{w}}([\mathbf{s}_0 : \mathbf{g}_\tau]) + \boldsymbol{\eta}, \quad \boldsymbol{\eta} \sim \mathcal{N}(\mathbf{0}, \sigma^2 \mathbf{I}) \quad (6)$$

Here,  $\mathbf{0}$  is a vector of all zeros,  $\mathbf{I}$  is the identity matrix, and  $\sigma^2$  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,

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

where  $\sigma_{\max}^2$  is a pre-defined maximum variance threshold, and  $\rho$  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.

---

#### Algorithm 1: Training One ForeSIT Agent

---

```

Randomly initialise  $\boldsymbol{\theta}, \mathbf{w}, \boldsymbol{\omega}$ 
Initialise replay buffer  $M = \emptyset$ 
 $\sigma^2 = \sigma_{\max}^2$   $\triangleright$  Imagination Noise
while  $episode < MAX\_EPISODE$  do
     $(\mathbf{s}_0, g_\tau) \sim E_{\text{RND}}$   $\triangleright E_{\text{RND}}$  is a random environment
     $\mathbf{I}_\tau = f_{\mathbf{w}}([\mathbf{s}_0 : \mathbf{g}_\tau]) + \boldsymbol{\eta}, \boldsymbol{\eta} \sim \mathcal{N}(\mathbf{0}, \sigma^2 \mathbf{I})$   $\triangleright$  Eq. (6)
    while  $a_t \neq STOP$  &  $t \leq T$  do
         $a_t \sim \pi_\theta(a | \mathbf{s}_t, \mathbf{g}_\tau, \mathbf{I}_\tau)$ 
    end
    Compute  $\alpha$  using eq. (4) for the trajectory
    Update  $\boldsymbol{\theta}$  and  $\boldsymbol{\omega}$  via eq. (8) and (9)
    if trajectory is successful in the environment then
         $\hat{\mathbf{s}}_\tau = \mathbf{s}_{t^*}, t^* = \arg \max_t \alpha_t$ 
         $M = M \cup \{\mathbf{s}_0, \mathbf{g}_\tau, \hat{\mathbf{s}}_\tau\}$   $\triangleright$  Update Buffer
    Update the average success rate  $\rho$ 
    if  $|M| = M_{\max}$  then
        for  $epoch \leq Epoch_{\max}$  do
            for  $\{\mathbf{s}_0, \mathbf{g}_\tau, \hat{\mathbf{s}}_\tau\} \in M$  do
                 $\boldsymbol{\omega} \leftarrow \boldsymbol{\omega} - \beta \nabla_{\mathbf{w}} |\hat{\mathbf{s}}_\tau - f_{\mathbf{w}}([\mathbf{s}_0 : \mathbf{g}_\tau])|$ 
            end
        end
         $M = \emptyset$ 
    Update  $\sigma^2$   $\triangleright$  Eq. (7)
end

```

---

Finally, we integrate the sub-goal selection and the imagination into the on-policy Asynchronous Advantage Actor-Critic (A3C) [17] algorithm which allows for efficient and parallel training of multiple agents. We devise the following revised objectives which integrates our ForeSIT into RL:

$$\mathcal{J}_\pi^*(a_t | \mathbf{s}_t, \boldsymbol{\theta}) = -\log \pi(a_t | \mathbf{s}_t, \mathbf{g}_\tau, \mathbf{I}_\tau; \boldsymbol{\theta})(r_t + \quad (8)$$

$$\gamma V_\theta(\mathbf{s}_{t+1}^*) - V_\theta(\mathbf{s}_t^*)) + \beta_H H_t(\pi)$$

$$\mathcal{J}_V^*(\mathbf{s}_t, \boldsymbol{\theta}) = \frac{1}{2}(V_\theta(\mathbf{s}_t^*) - R)^2 \quad (9)$$

We summarise the training of our approach in Algorithm 1. As may be observed, in the spirit of A3C, each agent keeps its own buffer and computes its success rate.

## 5. Experiments

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

### 5.1. Experimental Setup

We use the AI2THOR [13] environment to benchmark 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. We run our experiments on two distinct setups of this

Method	SPL	SR	SPL>5	SR>5
<b>First Setup</b>				
A3C [33]	14.68	33.04	11.69	21.44
A3C+MAML [33]	16.15 $\pm$ 0.5	40.86 $\pm$ 1.2	13.91 $\pm$ 0.5	28.70 $\pm$ 1.5
A3C+ForeSIT	<b>15.23</b> $\pm$ 0.4	<b>36.80</b> $\pm$ 1.1	<b>13.14</b> $\pm$ 0.3	<b>27.55</b> $\pm$ 1.4
A3C+MAML+ForeSIT	<b>16.75</b> $\pm$ 0.5	<b>45.5</b> $\pm$ 1.0	<b>15.8</b> $\pm$ 0.6	<b>34.7</b> $\pm$ 1.1
<b>Second Setup</b>				
A3C+ORG [5]	37.5	65.3	36.1	54.8
A3C+ORG+ForeSIT	<b>39.41</b> $\pm$ 0.3	<b>68.0</b> $\pm$ 0.6	<b>36.85</b> $\pm$ 0.4	<b>56.11</b> $\pm$ 0.8

Table 1. A quantitative comparison of our method against four baseline methods on two different environment setups. *SPL>5* and *SR>5* show the metrics for trajectories longer than 5 steps. Our approach improves all the commonly used evaluation metrics compared to two previous state-of-the-art methods [5, 33]. The two methods are significantly different in the methods used for modelling the state space.

simulator for fair comparison against previous state-of-the-art approaches. In both of these setups 20 different scene layouts of each room type are used for training; 5 scenes for validation and 5 for test. We provide the final results on the test set based on the best performing model on the validation set. For 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 [33, 35]. 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 ego-centric view of the agent.

In the second setup we follow the configuration recently introduced in [5]. 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<sup>3</sup>. 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 [2, 5, 33, 35] 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.

<sup>3</sup>We think due to the incorporation of an off-the-shelf object detector the list of objects in this setup is adjusted by the authors in [5] for more efficient detection.

## 5.2. Implementation Details

We use A3C [17] as the basis of our method. This actor-critic algorithm is a good fit for visual navigation task since the agents can explore in parallel and asynchronously, which is more computationally efficient. Our method can be readily integrated into any actor-critic method.

Our backbone model comprises of 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 [9] and the Glove embedding [10] 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 [5]. 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 ForeSIT’s  $f_w$  comprises 6 fully-connected layers that receive the concatenation of the state representation and the target object embedding as input. We use a tanh non-linearity 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 imagination separately. After the imagination module is trained, the weights are transferred to a model shared between the agents and the replay buffer emptied. For imagination noise we use  $\sigma_{max}^2 = 0.9$  and the last 100 episodes to compute the moving average for the success rate.

## 5.3. State-of-the-art Results

In Table 1 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

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

Table 2. Detailed comparison with 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 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 foresight is more important.

A3C RL algorithm. This is the main baseline that shows how a simple RL objective can perform without any extra components or modifications. **A3C+ForeSIT** is the variant of A3C with our ForeSIT module added. We see that empowering the agent with imagination improves the success rate by more than 3%. The improvement on longer-horizon tasks, *e.g.* longer trajectories, is even more significant, more than 6%. We conjecture that this is mainly because in longer trajectories the agent can forget the objective by focusing the resources on encoding the complex visual observations, or it is more likely to follow a trajectory that does not lead to the target. Adding our ForeSIT module, however, helps the agent both remember the target as well as taking a more confident trajectory by aiming for a sub-goal.

To evaluate the performance of our approach when updating the model in the test environment we utilise meta-learning similar to **A3C+MAML** [33]. This allows the agent to adapt to unseen environments at test time. **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+ForeSIT** 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+ForeSIT** on the SPL over the long trajectories implies that ForeSIT helps avoid futile wandering around.

The third baseline method that we compare against in the second setting is **A3C+ORG** [5]. The main contribution of the authors of this method is to use an off-the-shelf ob-

ject detector, *e.g.* FasterRCNN [23], and incorporate the detected object bounding boxes along with confidence scores by building a neural graph, inspired by [35]. 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.

However, in order to show that our ForeSIT module is general enough to work even in conjunction with an object detector we provide the results for **A3C+ORG+ForeSIT**. We observe that our method improves the success rate by over %2 compared to [5]. This shows that, irrespective of the quality of the state representations empowering the agent with imagination, using our ForeSIT module can help improve navigation performance. It also confirms the value of imagination for navigation, and the modularity and generalisability of our simple approach. Note that we do not further compare our results with the addition of imitation learning into this problem, as the authors did in [5]. This is because we are only concerned with adding imagination into RL in this paper.

#### 5.4. Ablation Studies

In this section we present experimental results that provide further insight into the contributions of the different components of ForeSIT.

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

In Table 3, we consider a random imagination by replacing the output of our ForeSIT with  $\tanh(s)$  where  $s$  is sampled from a Gaussian noise distribution, *i.e.*  $s \sim \mathcal{N}(\mathbf{0}, 0.5\mathbf{I})$ , shown in **Ours-RND**. The  $\tanh$  non-linearity assures that the noise is similar in value to the imagined states generated by ForeSIT, 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 imagined input to some extent, but not completely, hence the slight performance degradation.

We further compare our approach with the case of pre-

Method	SPL	SR
<b>A3C+ORG</b> [5]	37.5	65.3
<b>Ours-RND</b>	37.57	64.8
<b>Ours-INT</b>	37.78	63.8
<b>Ours-ATT</b>	37.76	65.4
<b>Ours-ForeSIT</b>	<b>38.66</b>	<b>67.6</b>

Table 3. Various ablation studies for our ForeSIT.

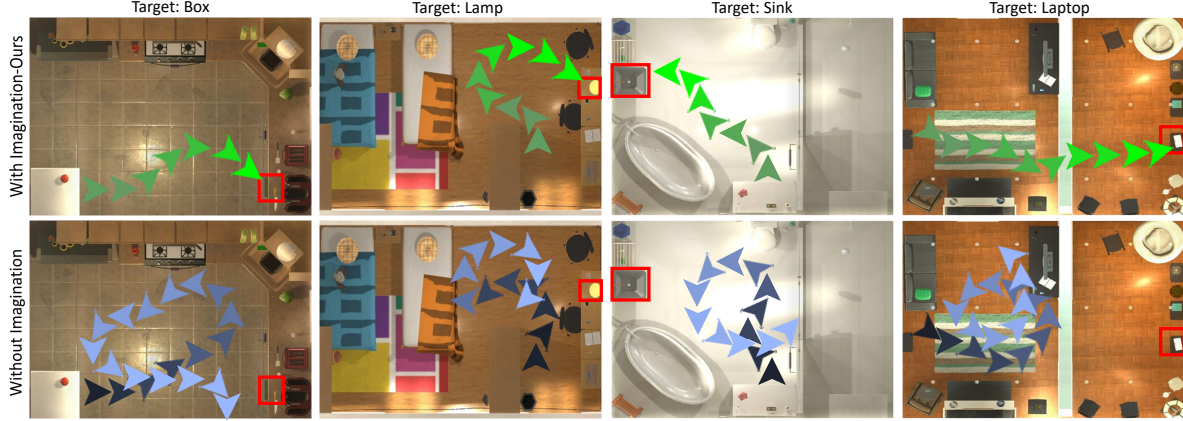


Figure 3. Qualitative trajectory comparison against the baseline method for varying room types and target objects. Our agent can efficiently navigate towards the target by imagining the sub-goal state.

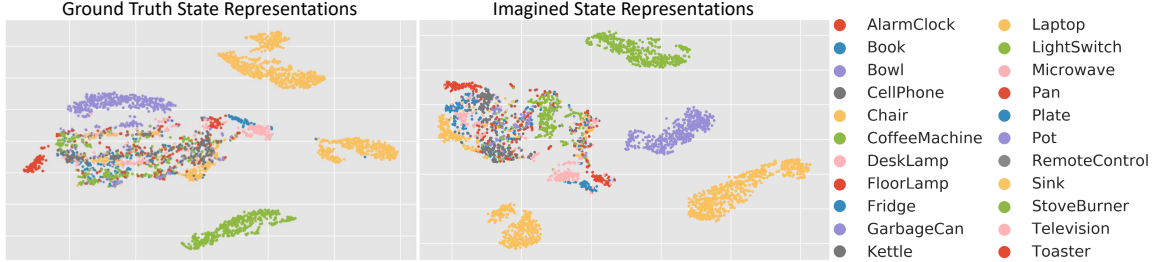


Figure 4. Comparison of the state representations generated using ForeSIT with the ground truth state representations. ForeSIT is conditioned on the target objects so it can learn the structure of the states efficiently.

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

**Imagination Intervals:** Rather than imagining the sub-goal from the initial state, we can consider imagining in multiple intervals, *i.e.* in a fixed interval and number of steps. In that case, in each interval 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 Imagination** It can be seen in Figure 4 that ForeSIT can accurately learn the structure in the state representations used by the agent’s state encoder. To further investigate the quality of the imagined states we train a Conditional Variational Auto-Encoder (C-VAE) [26] on the state representations. Then we use ForeSIT 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 imagination achieves a lower average loss, 0.012 vs 0.018, while being significantly less complex.

## 5.5. Qualitative Comparison

In Figure 3 we compare four sample navigation episodes between our method and the baseline method [33]. It can be commonly seen in all of these trajectories that ForeSIT enables the agent to navigate towards the target object more intelligently. The baseline method, however, has difficulty finding the correct path and stops after some wandering around.

## 6. Conclusion and Future Work

We have shown that it is possible to empower visual navigation agents with foresight by the addition of ForeSIT. This is achieved by imagining a future sub-goal in the latent space that leads to a successful navigation episode. Extensive experiments show how that ForeSIT can be integrated into a wide variety of methods to improve their navigation performance. While it has proven effective, we consider this work as an initial step towards enabling imagination for visual navigation. As the next steps we plan to



investigate the role of imagination enabling better exploration. We also intend to consider learning from failures by updating the imagined sub-goals that did not lead to success.

## References

- [1] 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. 2
- [2] 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. 6
- [3] Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, OpenAI Pieter Abbeel, and Wojciech Zaremba. Hind-sight experience replay. *Advances in neural information processing systems*, 30:5048–5058, 2017. 2
- [4] Howard Chen, Alane Suhr, Dipendra Misra, Noah Snaveley, and Yoav Artzi. Touchdown: Natural language navigation and spatial reasoning in visual street environments. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 12538–12547, 2019. 2
- [5] Heming Du, Xin Yu, and Liang Zheng. Learning object relation graph and tentative policy for visual navigation, 2020. 6, 7
- [6] 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. 1
- [7] David Ha and Jürgen Schmidhuber. World models. *arXiv preprint arXiv:1803.10122*, 2018. 3, 5
- [8] Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination, 2020. 3
- [9] 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. 6
- [10] Richard Socher, Jeffrey Pennington, and Christopher D Manning. Glove: Global vectors for word representation. *Cite-seer*. 3, 6
- [11] John Kanu, Eadom Dessalene, Xiaomin Lin, Cornelia Fermüller, and Yiannis Aloimonos. Following instructions by imagining and reaching visual goals, 2020. 2
- [12] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013. 2
- [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. 2, 5
- [14] Xiangyun Meng, Nathan Ratliff, Yu Xiang, and Dieter Fox. Scaling local control to large-scale topological navigation, 2019. 2
- [15] 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. 2
- [16] 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. 3
- [17] 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. 2, 4, 5, 6
- [18] 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. 2
- [19] 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. 2
- [20] 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. 2
- [21] Khanh Nguyen and Hal Daumé III. Help, anna! visual navigation with natural multimodal assistance via retrospective curiosity-encouraging imitation learning. *arXiv preprint arXiv:1909.01871*, 2019. 2
- [22] Yuankai Qi, Qi Wu, Peter Anderson, Xin Wang, William Yang Wang, Chunhua Shen, and Anton van den Hengel. Reverie: Remote embodied visual referring expression in real indoor environments. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9982–9991, 2020. 2
- [23] 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. 7
- [24] 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. 2
- [25] Manolis Sava, 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. 2

- [26] 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. 8
- [27] Yunhao Tang and Alp Kucukelbir. Hindsight expectation maximization for goal-conditioned reinforcement learning, 2020. 2
- [28] Jesse Thomason, Michael Murray, Maya Cakmak, and Luke Zettlemoyer. Vision-and-dialog navigation. In *Conference on Robot Learning*, pages 394–406, 2020. 2
- [29] 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. 4
- [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. 2
- [31] 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. 3
- [32] Erik Wijmans, Samyak Datta, Oleksandr Maksymets, Abhishek Das, Georgia Gkioxari, Stefan Lee, Irfan Essa, Devi Parikh, and Dhruv Batra. Embodied question answering in photorealistic environments with point cloud perception. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6659–6668, 2019. 2
- [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. 2, 3, 6, 7, 8
- [34] 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. 2
- [35] Wei Yang, Xiaolong Wang, Ali Farhadi, Abhinav Gupta, and Roozbeh Mottaghi. Visual semantic navigation using scene priors. *arXiv preprint arXiv:1810.06543*, 2018. 2, 3, 6, 7
- [36] 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. 1, 2, 3