Target-Driven Navigation with Imitation Learning

Jongho Kim
Stanford University
jkim22@stanford.edu

Seungbin Jeong
Stanford University
sjeong91@stanford.edu

Abstract

Reinforcement learning (RL) is an area of machine learning which concerns on how to train agents in order to achieve complex goals in uncertain and stochastic environments. Recently, combining deep learning techniques with RL has shown promising results. However, there are two well-known issues of Deep RL, which are (1) *the problem of specifying a suitable reward function for agent to optimize and* (2) *the problem of time complexity, i.e., the model requires multiple trial-and-error episodes for learning a behavior policy that indicates which actions to be selected in a fashion that correctly achieves goals.*

In this report, we address these issues and apply our model to target-driven visual navigation. To address these issues, we applied a novel approach, called imitation learning (IL), to sequential decision making based on RL framework. Particularly, we used the behavioral cloning approach, which enables the active agent to mimic an expert (or mentor) policy without usage of reward function. We show that our method (1) requires less training time than the state-of-the-art deep RL method with improved performance in navigation problem and (2) generalizes across targets.

1. Introduction

In recent years, supervised learning has shown great performances in many practical applications such as image classification and pattern recognition ($e.g.$ [16], [3]). However, as pointed out by Andrew Ng, [11], it is difficult to provide a large amount of proper training data in sequential decision making problems and this prevents it to be used in practice. For example, providing explicit supervision for a four-legged robot to walk is difficult since we do not know what the initial "correct" actions for walking.

Instead, we can train an agent to work correctly by letting the agent perform various actions and providing a relevant reward regarding the chosen action. By setting the main goal of the agent to maximize the reward and solve this optimization problem using learning algorithm, the agent can learn how to choose an action so as to obtain large rewards. This framework, called reinforcement learning (RL), is a common framework for solving sequential decision problems. In recent years, sequential prediction problems frequently arose in practice and several approaches used the RL framework generated a string of successful results. For instance, [5] proposes an RL based method which also incorporated deep learning technique and applied it to a quadrupedal robot system.

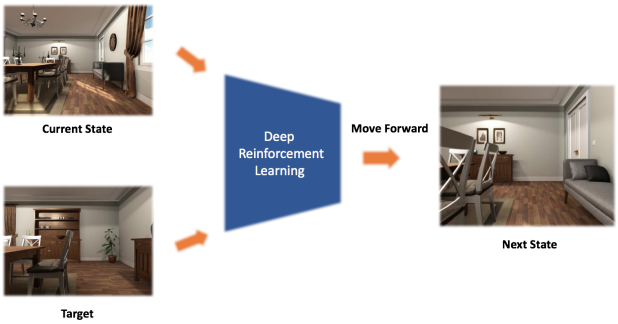

Figure 1: **Target-driven Navigation Decision Making**

An agent navigating a space in pursuit of a target based on its sight is a very representative problem of deep reinforcement learning (DRL). However, standard DRL models ($e.g.$, [8], [10]) address the problem by setting the main goal embedded within the learned model parameters because the policy learned can be used for determining actions for a variety of agent states, but the goal is still fixed and cannot be changed during application. Since training DRL agents again for a new task is computationally expensive, we introduce a *Target-driven* model which achieves a higher adaptability and flexibility by taking targets as inputs rather than hard-coded parameters [19]. In our model, the agent determines the next action based on both its current state and pursuing target, and the trained network will not be specific to a certain target used in the training. Figure 1 describes a simplified version of decision making process that our target-driven navigation model uses.

However, gathering large-scale action and interaction

data in real world is difficult with current image data collection methods, and still is a bottleneck in training the agent. Thus for training purpose, we use simulation frameworks with high-quality 3D scenes, called The House Of inteRactions (AI2-THOR) [19]. This framework is capable of collecting a large number of observations for action in various environment setting. Detailed discussion is written in **Methods** section.

One of common issues of DRL is the requirement of specifying a reward function. For each task, we want to provide a correctly measured reward once an agent chooses an action but this function is not deterministically known for every environment and goal settings. For instance, we can give -1 points for every step the agent makes without reaching the target, and give 100 points when it finally finds out the target. This heuristic method may not be optimal and it is not guaranteed that this approach leads for the agent to achieve the optimal policy, which ideally outputs the best action for every state and target. Since RL is very computation-expensive, trying a large number of hyperparameters to find out the best reward function is tedious.

To address this issue, we introduce imitation learning (IL) method into the field of RL. We apply the behavior cloning approach where we generate an expert policy and incorporates this policy during the process of training the agent. Given a current state-target pair, this policy outputs a set of optimal actions (or ground truth actions) to reach the goal. In other words, we form correct labels of actions for each state just like in supervised learning method. Using this scheme, we train an agent to mimic this policy and eventually learn the optimal policy.

We report that our IL method requires less training time than DRL methods and also generate improved performance in navigation problem. In terms of generating the expert, we use the A-Star (A^*) search algorithm to find the shortest path at each location of the environment. We evaluate our method by comparison of our IL based model, the DRL model of [19] and model that incorporates both IL and DRL technique for the following task: *Target generalization*; within the same environment (scene), the agent looks for a target it has never seen during training. Although the targets are not seen during the training, they might share common routes with trained targets and we evaluate on the knowledge transfer aspects of the trained agent.

2. Related Work

RL has been used and achieved a string of successful results in many applications. [5] proposes a policy gradient RL approach and show its implementations in the commercially available quadrupedal robot platform called *Sony Aibo robot*. [6] proposes RL based method to autonomous helicopter flight. [9] applies RL to sequential decision making in ATARI games. In recent years, RL methods that combine techniques from deep learning with RL has shown a string of successful results. [15] proposes a DRL approach that also incorporates Monte-Carlo tree search that beats the world champion in the game of Go. Due to promising results from these methods, deep RL has gained wide popularity.

Imitation learning (also called *learning from demonstrations*), which considers the problem of acquiring complex policies for sequential decision problems, also has a long history. Survey articles can be found in [1] and [14]. Two main branches within IL are (1) *behavioral cloning*, where supervised learning technique is combined so that demonstrations are directly used to learn a mapping from observations to actions (e.g. [13]) and (2) *Inverse reinforcement learning*, which is a problem of estimating an appropriate reward function with an assumption that the given demonstration consists of optimal actions (e.g., [12]). [2] shows how a reward function could involve feature learning and [18] shows that a reward function could be represented as a weighted combination of features. Since our work addresses behavioral cloning methods of IL and evaluate their performances, our works are closely related to [13].

Our work mainly focuses on the problem of navigating a space to find a given target using only visual inputs. There is a long history of works on navigation problems and here we present a brief overview. One of popular navigation methods is the map-based navigation method which requires a global map of the environment in order to make relevant decisions for navigation(e.g., [8]). Another class of navigation methods creates a map during the training phase and use it for navigation [17]. In addition, some methods involves humans guidances to build the map during a training phase [4]. In contrast to this class of navigation methods, there are *map-less* navigation methods as well, which have shown some promising results but focus on avoiding obstacles rather than pursuing a target.

Our work is *map-less* and is closely related to [19] which is categorized as a map-less navigation method of solving navigation problem. Many previous standard deep RL models focus on finding a direct mapping (a deep neural network π) from state representations s to policy $\pi(s)$ and these models suffers difficulties when changes in task goal or changes in the rule of the game are adjusted. For example, Luke et al [7] shows how the DRL-based Go-playing system can be vulnerable to minor changes in the game rule. One possible method to overcome this issue is proposed at Yuke et al [19]. They propose that their model overcomes this issue by learning how seek new targets without re-training. It takes two RGB image inputs, one being the current observation and the other being the target location. It uses an actor-critic network whose policy is a function of the goal as well as the current state. Motivated by this method, our model also takes two inputs ($i.e.$, current ob-

servation RGB image and target location RGB image) and model finds a suitable policy. Note that the model in [19] designed a reward function heuristically in order to find the minimum length sequence of actions that move an agent from its current location to a target. For example, the model proposed in [19] awards a goal-reaching reward (10.0) upon task completion and small time penalty (-0.01) as immediate reward. Instead of heuristically designing a reward function, our models uses behavioral cloning technique to find the optimal policy.

3. Methods

3.1. The AI2-THOR Framework

To train and evaluate our model, it is necessary to have a suitable framework that allows performing actions and showing corresponding results. In the navigation problem, once the agent chooses an action, the framework should not only clearly display the consequence of choosing the action but also allow the agent to receive the resulting information. Also, the framework should be a good representation of the real world in terms of physics of the environment because detailed representation can assist the generalization of our model into the real world applications. In addition, the framework should be capable of incorporating different types of environments because it is necessary for our model to be trained and tested in various environments.

To efficiently apply our model with different types of environments, we used The House Of inteRactions (AI2-THOR) framework, proposed in [19]. This framework is designed by combining a physics engine (*Unity 3D*) with a deep learning framework (*Tensorflow*) which has a detailed model of various scenes. Two main advantages of this framework are as follows: (1) Direct communications between the physics engine and the deep learning framework. (2) 3D scenes in this framework include delicate texture and lighting similar to the real world. Python APIs of the framework and detailed discussion on this framework can be found in [19].

3.2. Target-driven Navigation Model with Imitation Learning

3.2.1 Problem Statement

The main objective (goal) is to find the minimal sequence of actions that lead the agent from its current location to a target. Here we propose a deep RL model that takes two inputs: (1) the current observation as an RGB image and (2) the target as an RGB image. The output of the model is actions in 3D such as move forward.

Figure 2: **The AI2-THOR Framework**

3.2.2 Problem Formulation

Navigational decisions requires an understanding of the current positions, the target locations and environment information ($e.g.$, scene layout). Therefore, we propose the model that accounts and uses these requirements. The main objective of our target-driven model is to learn a policy function $\boldsymbol{\pi}$ which takes two inputs, a representation of current state s_t and a representation of target g and produces a probability distribution over the action space $\boldsymbol{\pi}(s_t, g)$.

$$a \sim \pi(s_t, g \mid \theta)$$

where θ are the model parameters. In terms of a finite set that contains navigation tasks ($g \in G$), g can be refered as an index for the right set of model parameters θ for each task.

3.2.3 Model

The neural network structure is exactly the same as that of the Q-learning based DRL model of [19], except for the fact that our model skips the very last layer to compute the value function. The figure 3 describes the common network structure.

The whole network can be divided into two different parts. The first half is called the *generic Siamese layers*, and the latter half is called the *scene-specific layers*. The generic Siamese layers are common to all scenes for training and testing, whilst the scene-specific layers have different parameters for each scene of the THOR environment. Hence this network is designed in attempt to separate parameters for observation image processing and determination of policy and value as much as possible.

The first set of layers in the generic Siamese layers are the ResNet-50. The ResNet-50 layers are pre-trained

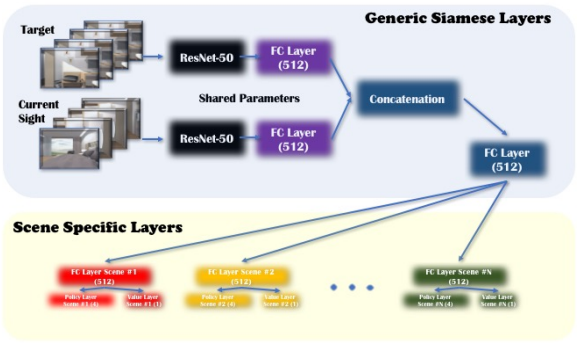

Figure 3: **Network architecture of our imitation learning model**

on ImageNet, and they produce 10 2048-d features on a 224x224x3 RGB image. Among these, we only make use of the first feature for each image to simplify the network. As the input to the network, we use 5 different images: the target observation and the four most recent agent observations(current state). The ResNet-50 features of the history of four recent agent observations are concatenated into a 8192-d vector, passed to a fully-connected layer, and projected into a 512-d vector. In the same fashion, 4 copies of the same target observations passed to the same structure with the shared parameters and result in a 512-d vector. Then those two 512-d vectors, each from the current state and target, are concatenated into a 1024-d vector and once more passed through a 1024x512 fully-connected layer.

Then the output of this fully connected layer is fed into different networks depending on the scene. These are the scene-specific layers, where the structure (number of layers and their sizes) is the same for all the scenes but their parameters differ. The first layer is 512x512 fully-connected layer. The second one is 512x4 if this is for the policy calculation, and 512x1 if this is for the value calculation. The original DRL model uses both of them, but our imitation model uses only the policy layer.

4. Dataset and Features

We describe the learning setup and dataset that we used for our model.

1. *Action space*: Mobile robots in real world have to function well with a number of mechanics. Although incorporating detailed mechanics into the mobile robots are significant part of real world applications, it makes difficulties in terms of learning. Therefore, we use AI2-THOR framework to handle underlying physics of 3D environment and train our model with four command-level actions: turn left, turn right, move forward, move backward. We use 0.5 meters of constant step length and 90 degree of turning angle. Each rotation and forward/backward movement by the agent is counted as one step.

2. *Observations and goals*: Both observations and goals are RGB images as agent's first person view. Given a target image, agent's objective is to navigate to the location where the target image is taken.

3. *Scene data*: We used 6 amongst the 20 scene data dumps provided by THOR challenge [20]. The data includes 5 kitchen scenes, 5 living rooms scenes, 5 bedroom scenes and 5 bedroom scenes, and we used following files: `bathroom_02`, `bathroom_03`, `bedroom_03`, `bedroom_04`, `living_room_03`, and `living_room_08`

4. *Optimal Policy Generation*: One of the most difficult problems in using supervised learning in robotics or navigation problems is the generation of proper training set. The very first step to make a reasonable training set is to have an optimal policy function for enough number of possible states. Procedure to generate the optimal policy is following:

    1. Accumulate the observation images and label them with location ID number
    2. State-action transition graph (G) where G_{ij} indicates the location ID of destination if action j is taken at location i.
    3. Calculate shortest number of steps from location i to location j (for all locations) and make it into a square matrix, S.
    4. Generate the optimal policy $\pi(s_t, g)$ that outputs which action to take when two inputs current location ID s_t and destination ID g are specified.
    5. Generate training input image set using the optimal policy

    Data for step (1), (2) and (3) were provided by scene data from Allen Institute. Using the graph G and shortest distance matrix, S, we obtained the optimal policy.

5. *Training Data Input Generation*: Standard deep RL models use history frames to learn a policy at the current state. Yuke et al [19] used concatenation approach on 4 history frames to account for the agent's past motions. Our model, expecting the feasible mixture of the DRL and the IL, used the same network as the DRL. However, since the optimal action in principle is determined solely by the current state and the destination in the supervised learning, what to be input as the history frames of the agent is ambiguous. Our first try was to put zero features as the input to the history frames. However, that led to a strong discrepancy when actual

features were input. Then we tried to input random features as history frames, only to fail again. Realizing that those methods could not emphasize the significance of the target and current observation over history frames, we chose to use two input sets simultaneously: (1) a training set that has all four history frames equal to the current frame (2) a training set with four history frames assuming that the agent is following the optimal path. Our intuition is that (1) would urge the agent to avoid getting stuck in a certain state, and (2) would assist the agent to follow the optimal path once it enters it. Together, they would give emphasis on the current observation and the target.

5. Experiments

The main task of a navigation problem is to move from the current location to the target with minimal time and effort. We evaluate our model with the target-driven navigation model of [19] as a baseline model. We measure and plot the average trajectory lengths in evaluation versus training time for both our imitation model and the baseline model. From this we can compare the training time-efficiency of two models. The evaluation is done for different targets within the same scene (environment), which were unseen during the training, to check the target generalization.

There were 6 different training methods for comparison: (1) One-hour training using DRL, (2) one-hour training using IL, (3) two-hour training using DRL, (4) two-hour training using IL, (5) one-hour DRL followed by one-hour IL, and (6) one-hour IL followed by one-hour DRL.

For each scene amongst 6 scenes, we randomly chose 5 targets guaranteed not to have been seen in any training. For each target, we chose 100 random starting state of the agent, and measured the average steps the agent took before reaching the target. The maximum, minimum average steps as well as their standard deviations were recorded.

Since different scenes had different complexities and map scales (possible combination of locations and rotations), we divided each number(maximum steps, etc.) of the scene with the corresponding number of the one-hour DRL.

Table 1 shows the maximum, minimum. average steps and the standard deviations normalized in such a way that those numbers for the one-hour training results are 100 steps.

One hour training of IL showed an inferior performance compared to the 1-hour DRL. It showed 182% increased average steps, and extremely high instability of performance from target to target. Hence, it implies that with short training IL is weak against over-fitting.

However, the imitation learning showed a much improved performance when we increased the length of training. We trained our IL model for two-hour and it showed some promising results. It reduced the average number of steps required by a factor of 4. The performance showed much improvements than that of DRL model proposed by Yuke et el [19]. The average steps of model proposed by Yuke was 98.5 steps but our model was 71.6 steps. Although our model achieves higher maximum steps than that of the DRL model, our model can accomplish certain tasks with much small number of steps as the minimum steps column in the table indicates.

	Maximum Steps	Minimum Steps	Average Steps	Standard Deviation
DRL 1 hour	100.0	100.0	100.0	100.0
IL 1 hour	239.6	38.4	282.2	447.7
DRL 2 hours	115.9	81.6	98.5	147.9
IL 2 hours	135.6	20.1	71.6	203.6
DRL 1 hour+ IL 1 hour	274.5	24.4	290.2	609.9
IL 1 hour+ DRL 1 hour	93.2	80.6	112.8	99.2

Table 1: **Performance of IL Methods AND Other Models**

However, the mixture of the two methods did not show good performance. The imitation learning followed by the DRL showed almost 3 times worse average steps as well as 6 times worse standard deviation than those of the one-hour DRL.

Training the agent in the reversed order in such a way that DRL follows IL, the average steps were a bit higher but the standard deviation was a bit lower. Statistically this was not very different from training the agent just for an hour using DRL. For short-term trainings, the DRL and IL seemed to be incompatible with each other even though they shared exactly the same network structure.

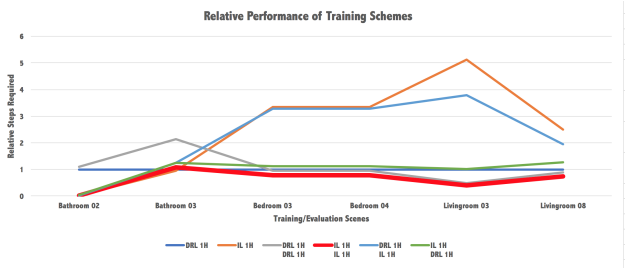

Figure 4: **Target Generalization**

Figure 4 shows the average steps normalized to one-hour DRL training for each scene used for training. Note that lower height implies shorter steps for navigation and red colored graph is our two-hour IL trained model.

One interesting thing to point out is that, each train-

ing methods showed its weaknesses at certain tasks. We stopped the evaluation when the agent failed to find the target within 10,000 steps and defined this as a failure of the trial. When more than 50 percent of the trials fails, we defined this as a failure of that certain scene-target pair. For example, two-hour IL method failed in task (target) 8 in the `living_room_03`, and two-hour DRL failed in task 130,140,150,160 in the `bathroom_02`. Each training methods totally failed for at least one of the evaluation tasks. This implies that, regardless of the training method, the small size of the network itself causes the lack of generalization.

6. Conclusion

We proposed an imitation learning (IL) based DRL framework for target-driven visual navigation. We addressed the problem of specifying a reward function for agent to optimize and the problem of requiring multiple trial-and-error episodes for the agent to learn a proper policy in RL. We compared our model with the state-of-the-art DRL methods proposed by [19] and two combined models that incorporates both IL and state-of-the-art DRL methods. Our experiments showed that our method achieves less number of steps for navigation tasks in comparison to other DRL models. We also showed that our model requires less training time in AI2-THOR framework.

We plan to investigate on why the mixture of two training methods led to a catastrophic results. If those two methods are trying to make the network resemble the same optimal policy, the ultimate network parameters should converge in the same way. However our results showed otherwise. Two possible reasons for this are (1) the sequential learning (DRL) innately leads to different parameters than parallel learning (IL). (2) The choice of observation history in our model was not compatible. If there is fundamental disparity between the parameters learned from two different methods, then we can completely discard the history layers to simplify the network in our model. If not, finding the best ways to feed in the artificial history would be a key to mixing different training methods with the same network structure.

Our work and results can lead to many potential extensions. We can train and test our model on environments (3D scenes) that requires longer distances ($e.g.$, 100 number of steps). Also, we can extend the action spaces ($e.g.$, add diagonal movements or yaw/pitch/roll) and build models that also learns and performs physical interactions in the framework. For example, instead of setting the target location as input, we can extend our model to possibly accept a particular object as input and train our model to navigate to the location where the object is.

7. Acknowledgements

We would like to thank Yuke Zhu for his helpful comments, advices and providing us the AI2-THOR framework for this project.

References

[1] B. D. Argall, S. Chernova, M. Veloso, and B. Browning. A survey of robot learning from demonstration. *Robot. Auton. Syst.*, 57(5):469–483, May 2009.

[2] C. Finn, S. Levine, and P. Abbeel. Guided cost learning: Deep inverse optimal control via policy optimization. *CoRR*, abs/1603.00448, 2016.

[3] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[4] K. Kidono, J. Miura, and Y. Shirai. Autonomous visual navigation of a mobile robot using a human-guided experience. *Robotics and Autonomous Systems*, 40(23):121 – 130, 2002. Intelligent Autonomous Systems - {IAS} -6.

[5] N. Kohl and P. Stone. Policy gradient reinforcement learning for fast quadrupedal locomotion. In *Proceedings of the IEEE International Conference on Robotics and Automation*, May 2004.

[6] T. Kollar and N. Roy. Trajectory optimization using reinforcement learning for map exploration. *Int. J. Rob. Res.*, 27(2):175–196, Feb. 2008.

[7] B. M. Lake, T. D. Ullman, J. B. Tenenbaum, and S. J. Gershman. Building machines that learn and think like people. *CoRR*, abs/1604.00289, 2016.

[8] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. P. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu. Asynchronous methods for deep reinforcement learning. *CoRR*, abs/1602.01783, 2016.

[9] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. A. Riedmiller. Playing atari with deep reinforcement learning. *CoRR*, abs/1312.5602, 2013.

[10] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 02 2015.

[11] A. Ng. Reinforcement learning and control. 2016.

[12] A. Y. Ng and S. J. Russell. Algorithms for inverse reinforcement learning. In *Proceedings of the Seventeenth International Conference on Machine Learning*, ICML '00, pages 663–670, San Francisco, CA, USA, 2000. Morgan Kaufmann Publishers Inc.

[13] S. Ross, G. J. Gordon, and D. Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In G. J. Gordon, D. B. Dunson, and M. Dudk, editors, *AISTATS*, volume 15 of *JMLR Proceedings*, pages 627–635. JMLR.org, 2011.

[14] S. Schaal. Is imitation learning the route to humanoid robots?, 1999.

[15] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587):484–489, Jan. 2016.

[16] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. E. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. *CoRR*, abs/1409.4842, 2014.

[17] D. Wooden. A guide to vision-based map building. *IEEE Robotics Automation Magazine*, 13(2):94–98, June 2006.

[18] M. Wulfmeier, P. Ondruska, and I. Posner. Deep inverse reinforcement learning. *CoRR*, abs/1507.04888, 2015.

[19] Y. Zhu, R. Mottaghi, E. Kolve, J. J. Lim, A. Gupta, L. Fei-Fei, and A. Farhadi. Target-driven visual navigation in indoor scenes using deep reinforcement learning. *CoRR*, abs/1609.05143, 2016.

[20] Y. Zhu, R. Mottaghi, E. Kolve, J. J. Lim, A. Gupta, L. Fei-Fei, and A. Farhadi. Target-driven visual navigation in indoor scenes using deep reinforcement learning. In *International Conference on Robotics and Automation (ICRA)*, 2017.