DEEP REINFORCEMENT LEARNING FOR TARGET-DRIVEN NAVIGATION IN INDOOR SCENES

VASANTH KOLLI [VASANTHK@SEAS.UPENN.EDU], BHARATHRUSHAB MATHRIPRAGADA [MBHARATH@SEAS.UPENN.EDU],
SHIVANI REDDY RAPOLE [SRAPOLE@SEAS.UPENN.EDU],

ABSTRACT. In this project, we trained a Reinforcement Learning model that can perform navigation in indoor scenes based on visual targets and observations. We base our work on the paper 'Target-driven Visual Navigation in Indoor Scenes using Deep Reinforcement Learning'. We use indoor scene dumps simulated by AI2THOR as discretized images and actions to re-implement the entire architecture from scratch in PyTorch in single−threaded environment. We observe the performance of the trained model and the generalization performance of the pre-trained model.

1. INTRODUCTION

In Robotics, visual navigation is an essential aspect of a broad range of applications starting with mobile robotics to autonomous driving. Approaches to solve this problem statement are becoming feasible with the availability of highperforming computing hardware and cheaper cameras. In this project, we take up the task of target-driven visual navigation basing our work on the paper - 'Target-driven Visual Navigation in Indoor Scenes using Deep Reinforcement Learning' [1]. The initial state and the final target state are given as image inputs (2D images) to the deep RL model which tries to predict the minimum length sequence of actions that move an agent (in 3D space) from the initial to target state. It is important to note that the dataset we used is a set of simulated images from the AI2THOR framework that generates high quality 3D−scenes which are state of the art for indoor navigation tasks.

2. BACKGROUND

One of the major tasks of Robotics is to understand the correlation between the actions of an agent and the resulting effect on the environment. In the recent times, Deep Learning and RL−based methods have been gaining popularity in addressing those problem statements [2]. One such problem statement is Visual Navigation in indoor scenes. In this domain, the deep RL models are generally designed to reach a particular target state from any given starting point as input [3]. So, the model is expected to be trained for every target which is very time consuming and computationally expensive. Deep Reinforcement Learning models generally also require a large amount of expensive training episodes before they finally converge which indicates that the algorithms are using teh data inefficiently. This along with the issue of not being able to generalize to new target goals make deep RL infeasible to be applied to real world scenarios.

The target-driven model introduced in our base paper works completely based on visual inputs and targets. Since the proposed RL model takes a target image as input, retraining for every different target is not required anymore. The model now trains and learns the policy based on the current state and the final target rather than depending just on the current state and embedding the the fixed target/goal information in the model itself in the form of model weights.

The intuition behind designing such an approach is to exploit a few known facts like:

- Different training episodes for different targets try to explore similar paths in the scene. Therefore if we try to train for various targets together, these episodes will try to share information amongst each other.
- Different scenes also share a few spatial/semantic relation aspects in a scene. e.g. fridge is generally placed near the microwave, dining chairs are near the table, etc.
- It is easier to learn to navigate to new targets when the model is generalized over other targets in training.

3. RELATED WORK

There have been a lot of approaches for visual navigation in the recent research. Mapbased navigation methods will need the global map to make any kind of decisions [4, 5]. An improved version of methods generate this map on the fly and have a human phase to build it [6, 7]. Our model is mapless, it doesn't require any map of the environment. There are a few existing research on developing learning methods that help in generalizing to multiple targets [8, 9].

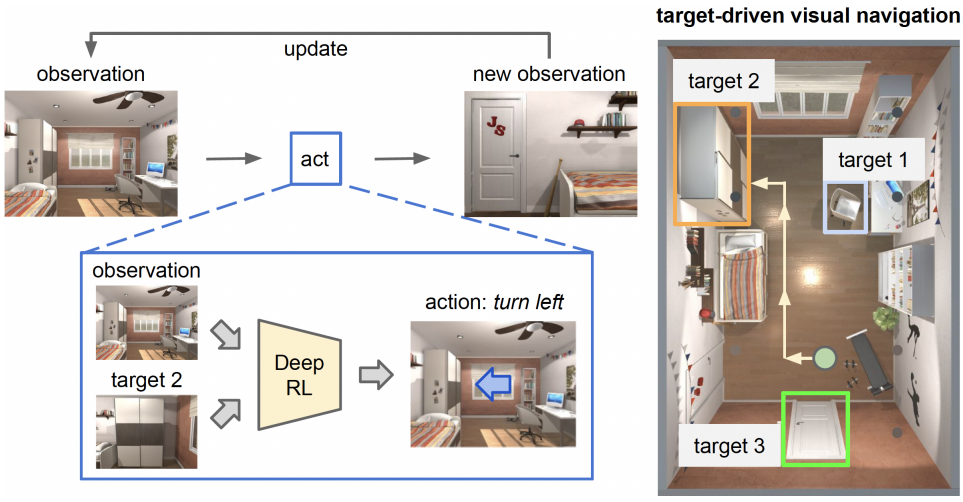

FIGURE 1. Goal of our Deep RL model - Navigate from the initial observation/input to the final target in minimum number of steps.

Physics engines have also been used to learn the realworld scene dynamics based on images [10, 11]. But, in this work retraining is not required since the target is a part of the input. Hence, it can also be generalized to realworld images as well.

4. APPROACH

4.1. **Dataset Description:** For this project, we use the images simulated by the framework AI2THOR [12]. This is designed by integrating the physics engine Unity 3D with a deep learning framework. AI2THOR was built by providing a few reference images to artists who were asked to create 3D scenes with lightning and texture similar to the reference image. It has 32 scenes which contain 68 objects on an average and 4 scene types - bedroom, living room, kitchen, bathroom. This particular framework generates synthetic images that mimic the real world images as much as possible. This specific aspect makes the model generalize better over real world scenarios as well.

Here, instead of using the AI2 Thor framework we are using scene dumps which are discretized version of the AI2 Thor indoor scenes.

4.2. **Learning setup:** We'll first define the main components of our Reinforcement Learning setup

- Action Space: For this indoor navigation task, there are four possible actions - turning right, turning left, moving backward, moving forward. The step length is 0.5 meters and the turning angle is 90°
- Observations and Goals: The goal and observation inputs are 2D RGB images in first person view. The images serve as an observation produced from the true state of the agent which is the position of the agent in the scene. The objective is to navigate from any initial position to goal position specified by an image target
- Reward design: The aim is to minimize the length of the trajectory to the target. So, we set a reward of +10.0 if it reaches the goal position and a small penalty of -0.01 for each time step.

4.3. **Model Architecture:** Our deep neural network is an approximation of nonlinear function that takes the images of current observation and the final target at time t as inputs and predicts the next action. Such navigation problems require a thorough understanding of relative distances and object positions between current and targets points. The deepsiamese actor critic model displayed in figure 2 captures a holistic sense of the scene. We first project both the observation and target into a common embedding space using siamese networks so that the spatial relations between them are preserved. These two projections are then combined into a common representation. This is then passed to a set of scenespecific layers which capture the specific characteristics of scenes. The model as a whole is an Actor-Critic model.

4.4. **Implementation:** AI2 Thor was the the simulation environment used in the original paper. However the simulation environment demands heavy compute. Since the action space is discrete, the entire scene can be discretized based on the possible actions and their corresponding states. Thus we instead opted to use a discretized indoor environment.

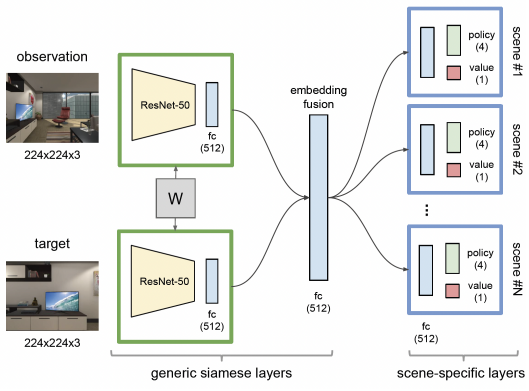

FIGURE 2. Workflow of the deep siamese actor-critic model

This was made available in the form of scene dumps of the original AI2 Thor indoor scenes which brought down the computational load for the simulation environment. The scene dumps can be readily used for training an agent, they further provide an additional functionality of visualizing the scenes in a first person view, which could be used for exploring the environment manually as we lack a map of the scene.

The original model was implemented in Tensorflow. The model architecure is shown in figure 2. We reimplemented the model from scratch in PyTorch without the multi threading component. We used an actor critic model to train on one of the scenes from the scene dumps mentioned above. The training was done for single target image with different initial positions in the scene. We trained for 750,000 frames. This training was not adequate and did not produce desirable results. The training was not sufficient as we observed that the agent rarely moved and instead would rotate about its fixed position to observe the various images from the environment.

Due to the limited compute resources and time, training the model for the recommended amount of 10 Million Frames was not feasible. We picked a checkpoint model from the original implementation which was trained on 1.6 Million frames and trained it further for 500,000 Frames. This too didn't produce satisfactory results although better than the model we trained for 750,000 frames.

We thus moved our focus to the original model trained with 10 million frames to evaluate its performance. We observed that the model trained for 10M frames in parallel on different scenes produced better results for the trained scenes but didn't generalize well.

The Actor critic loss with entropy regularization used by our model is defined as follows:

$$
\begin{aligned}
\textit{Entropy} : H\left(\pi\left(\cdot \mid s_t\right)\right) &= -\sum_{a \in A} \pi\left(a \mid s_t\right) \log \pi\left(a \mid s_t\right) \\
\textit{Advantage} : A\left(s_t, a_t\right) &= r_{t+1} + \gamma V\left(s_{t+1}\right) - V\left(s_t\right) \\
\textit{Critic Loss} &= MSE\left(V\left(s_t\right), r_t + \gamma V\left(s_{t+1}\right)\right) \\
\textit{Actor Loss} &= \left(\sum_{t=0}^{T-1} \log P_{\pi_\theta}\left(a_t \mid s_t\right)\right) A\left(s_t, a_t\right) \\
\textit{Total Loss} &= \textit{Actor Loss} + \textit{Critic Loss} - \beta * \textit{Entropy}
\end{aligned}
$$

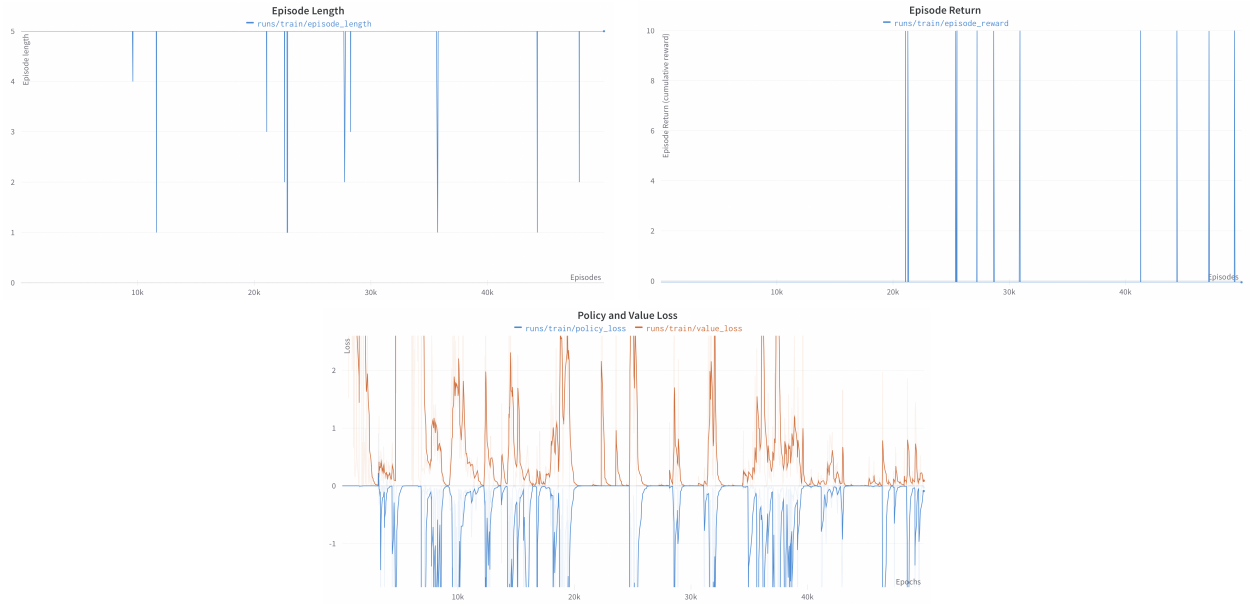

FIGURE 3. Episode Length, Episode Return and Loss curve for the initial 50,000 episodes of traning

5. EXPERIMENTAL RESULTS

We can observe from the loss plots in training that the loss over time is reducing and gradually we see episodes with a return close to +10 indicating that the agent is learning albeit slowly. However we observed that the learning is extremely slow and the agent did not perform well even when trained for 750,000 frames on a single target. The agent is able to navigate to the target if it is initialized very close to target but other wise the agent is unable to navigate to the target. Surprisingly the agent does not even move from the initial position in this case and keeps looking around observations without moving forward/backward. This is because we don't incentivize the movement of the agent directly as taking a step and rotating left/right, and since it has not yet learned enough it does not move anticipating better observations which it will never receive. Incentivizing motion early on and slowly annealing it later on could enable the model to learn faster and produce better results.

We analyzed the generalization performance of the original trained model over varying scenes and targets. We tested the agent to navigate to the same target for which it was trained in the same training scene. Here, we observe that the model is very efficient and navigates to the target with the shortest path possible from any randomly initialized starting position as seen in Fig 4. We tested the ability of the model to generalize a scene by initializing different targets in the scene where it was trained. Here the scenes are a distance of 0.5 meter (1 unit) or 1 meter (2 units) away from a trained target. Here we observe that the performance of he model deteriorates but it is able to navigate to the target occasionally as seen in Fig 5. Then we test the performance of the agent on scenes other than the training scene. Here we observed that the model fails to reach the target in any of the scenes as seen in Fig 5. Thus the model is unable to generalize outside the scene it was trained on.

```
bedroom_04.h5
terminal state id: 134

               num_iter : 0
               episode_length : 5
               episode_reward : 9.96
               inital state to target state shortest dist: 5
               final state to target state shortest dist: 0

               num_iter : 1
               episode_length : 2
               episode_reward : 9.99
               inital state to target state shortest dist: 2
               final state to target state shortest dist: 0
```

```
average scene length = 11.92
average scene reward = 9.8908
```

FIGURE 4. Trained scene and Trained targets, (Left) agent following shortest path to the target, (Right) average episode length and reward for 20 episodes

```
dist from target : 1
average episode length = 2679.4
average episode reward = -19.453333333337138
dist from target : 2
average episode length = 8002.866666666666
average episode reward = -78.02666666667807
```

```
bedroom_04.h5
terminal state id: 197

                num_iter : 0
                episode_length : 10002
                episode_reward : -100.02000000001426
                inital state to target state shortest dist: 17
                final state to target state shortest dist: 17

                num_iter : 1
                episode_length : 4
                episode_reward : 9.97
                inital state to target state shortest dist: 4
                final state to target state shortest dist: 0

                num_iter : 2
                episode_length : 1
                episode_reward : 10.0
                inital state to target state shortest dist: 1
                final state to target state shortest dist: 0

                num_iter : 3
                episode_length : 10002
                episode_reward : -100.02000000001426
                inital state to target state shortest dist: 9
                final state to target state shortest dist: 3

                num_iter : 4
                episode_length : 14
                episode_reward : 9.87
                inital state to target state shortest dist: 12
                final state to target state shortest dist: 0
```

```
scene name:kitchen_02
average scene length = 10002.0
average scene reward = -100.02000000001426


scene name:bathroom_02
average scene length = 10002.0
average scene reward = -100.02000000001426


scene name:living_room_08
average scene length = 10002.0
average scene reward = -100.02000000001426
```

FIGURE 5. Performance deteriorates on trained scene with different targets, (Top) Agent is unable to navigate efficiently to targets at a dist of 1 unit (0.5 m) and 2 units (1 m) away from trained targets, (Middle) Agent able to navigate to target from certain starting positions but not to others, (bottom) Agent unable to navigate to any target in a different scene (Note: episode terminates if agent unable to reach target in 1000 steps)

6. DISCUSSION

The experimental results evidently show that our model is not able to generalise well on a scene different from the one the model trained on. If we had more time and sufficient compute we could train the model on multiple threads parallelly for different scenes. This would help the model learn and generalise well on different scenes. We also want to compare our model with state of the art model based RL agent - Dreamerv2 to see how well our model performs. The model based agent is preferable for the task as the state of the scene doesn't change too much over time and the model based method would enable the agent to model the dynamics of the scene along with the spatial representations which would add to the generalisation performance/ability of the agent.

REFERENCES

[1] Yuke Zhu et al. *Target-driven Visual Navigation in Indoor Scenes using Deep Reinforcement Learning*. 2016. DOI: 10.48550/ARXIV.1609.05143. URL: https://arxiv.org/abs/1609.05143.

[2] Sergey Levine et al. *End-to-End Training of Deep Visuomotor Policies*. 2015. DOI: 10.48550/ARXIV.1504.00702. URL: https://arxiv.org/abs/1504.00702.

[3] Volodymyr Mnih et al. "Asynchronous Methods for Deep Reinforcement Learning". In: (2016). DOI: 10.48550/ARXIV.1602.01783. URL: https://arxiv.org/abs/1602.01783.

[4] J. Borenstein and Y. Koren. "Real-time obstacle avoidance for fast mobile robots". In: *IEEE Transactions on Systems, Man, and Cybernetics* 19.5 (1989), pp. 1179–1187. DOI: 10.1109/21.44033.

[5] J. Borenstein and Y. Koren. "The vector field histogram-fast obstacle avoidance for mobile robots". In: *IEEE Transactions on Robotics and Automation* 7.3 (1991), pp. 278–288. DOI: 10.1109/70.88137.

[6] Robert Sim and James J. Little. "Autonomous vision-based exploration and mapping using hybrid maps and Rao-Blackwellised particle filters". In: *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2006, pp. 2082–2089. DOI: 10.1109/IROS.2006.282485.

[7] E. Royer et al. "Outdoor autonomous navigation using monocular vision". In: *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2005, pp. 1253–1258. DOI: 10.1109/IROS.2005.1545495.

[8] Andrei A. Rusu et al. *Policy Distillation*. 2015. DOI: 10.48550/ARXIV.1511.06295. URL: https://xn--arxiv-nra.org/abs/1511.06295.

[9] Emilio Parisotto, Jimmy Lei Ba, and Ruslan Salakhutdinov. *Actor-Mimic: Deep Multitask and Transfer Reinforcement Learning*. 2015. DOI: 10.48550/ARXIV.1511.06342. URL: https://arxiv.org/abs/1511.06342.

[10] Ilker Yildirim et al. "Galileo: Perceiving physical object properties by integrating a physics engine with deep learning". In: (Dec. 2015).

[11] Roozbeh Mottaghi et al. *Newtonian Image Understanding: Unfolding the Dynamics of Objects in Static Images*. 2015. DOI: 10.48550/ARXIV.1511.04048. URL: https://arxiv.org/abs/1511.04048.

[12] Eric Kolve et al. *AI2-THOR: An Interactive 3D Environment for Visual AI*. 2017. DOI: 10.48550/ARXIV.1712.05474. URL: https://arxiv.org/abs/1712.05474.