TARGET DRIVEN VISUAL NAVIGATION: PYTORCH IMPLEMENTATION

Austin Harris

Ranveer Singh

AGENDA

Task

Environment

Method

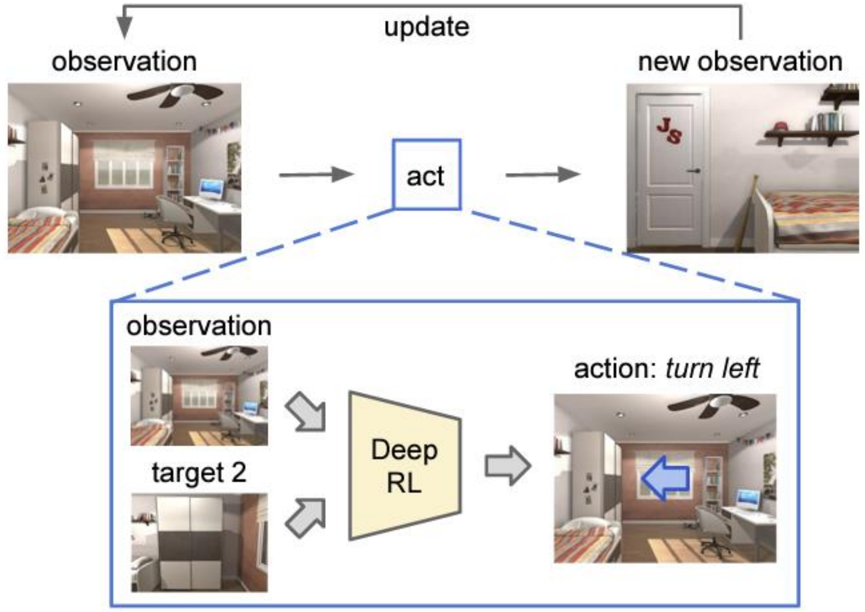
Training and Experiments

Conclusion

TASK

TASK

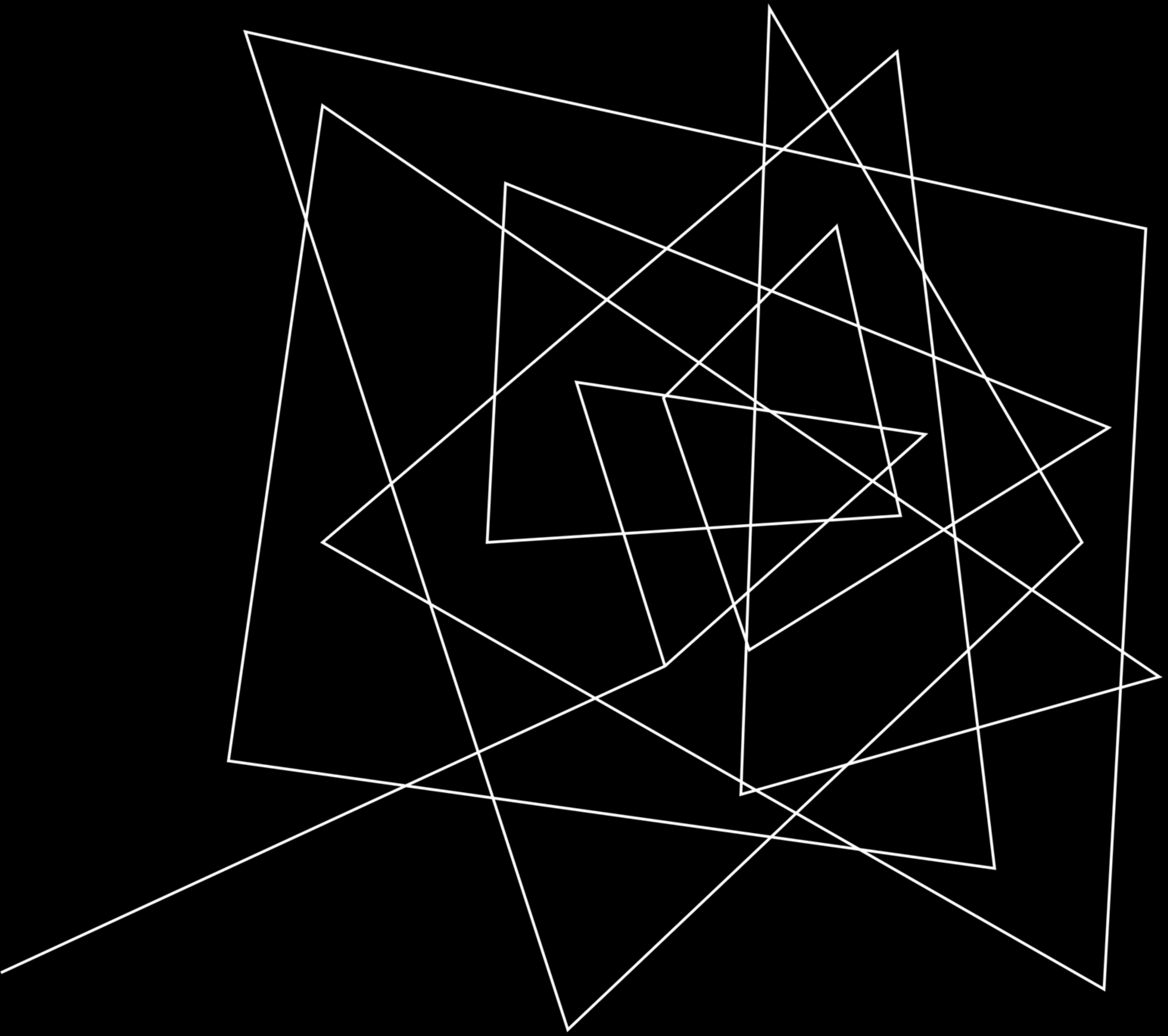
1. Our project is an implementation of the Target Driven Indoor Navigation using Reinforcement Learning (Yuke Zhu et al).
2. We implement an actor-critic model whose policy is the function of the goal as well as the current state for target driven visual navigation
3. The agent is given an image of the environment as a target and its goal is to navigate to where the image was taken in the least number of steps



GOAL: NAVIGATE TOWARDS THE VISUAL TARGET WITH MINIMUM NUMBER OF STEPS

The agent learns to navigate from its starting position to the target

In this case, the agent learns to navigate to the small table



ENVIRONMENT

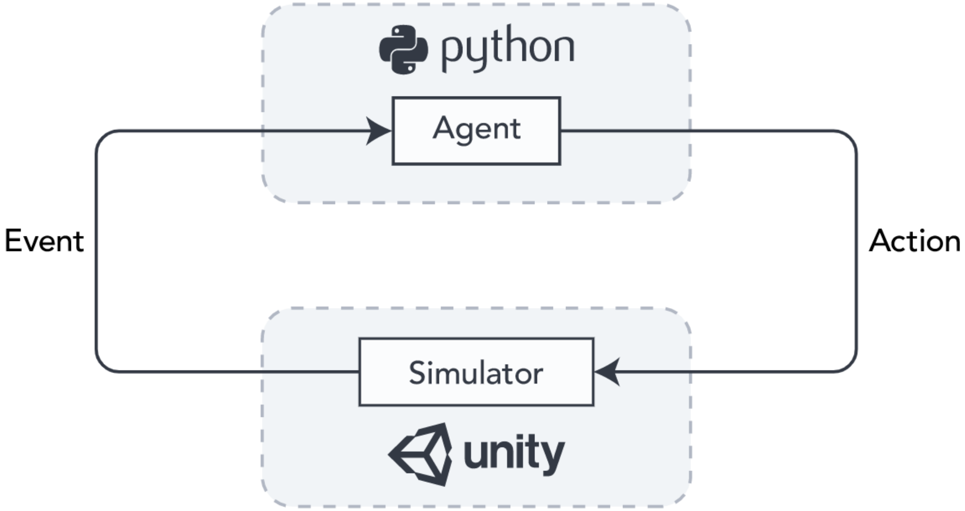
ENVIRONMENT

1. The original paper we are implementing, introduced the AI2-Thor environment, which we would be using for our work as well

2. AI2-Thor is a simulation environment built with Unity that enables agents to interact with a 3D environment
   .
3. Images from the physics engine are streamed to the network, and the network generates control commands for the engine to execute.

4. It is built with a python API which makes it ideal for use with Deep Reinforcement Learning

5. We would be testing our agent on various AI2-Thor environments, across different targets



EXAMPLE: FLOOR PLAN 212

- For our experiment, the agent only has access to the agent’s view and a target image
    - 300 * 300 RGB image for both observations and target
- The agent has no explicit knowledge to the floor layout

Agent View

Floor Layout

METHOD

Method: Deep Reinforcement Learning

We try to solve the problem using Deep Reinforcement Learning Problem

In the following slides, we define the

1. Action Space
2. Observation Space
3. Rewards
4. Agent Architecture

Action Space

1. Action space defines the actions our agent can take
2. We have four actions for our agent
    a. Move Ahead (Moves ahead by 0.5 m)
    b. Move Back (Moves back by 0.5 m)
    c. Rotate Right (Turns right by 90 degrees)
    d. Rotate Left (Turns left by 90 degrees)

```python
def actions(self):
    return ["MoveAhead", "MoveBack", "RotateRight", "RotateLeft"]
```

Observation Space

1. The observation space refers to the perception of the agent
2. For our implementation, both the current observation and target observation are 300 * 300 RGB images from AI2-Thor

Rewards

Since our goal is to minimize the number of steps take from starting position to goal, we define our reward as

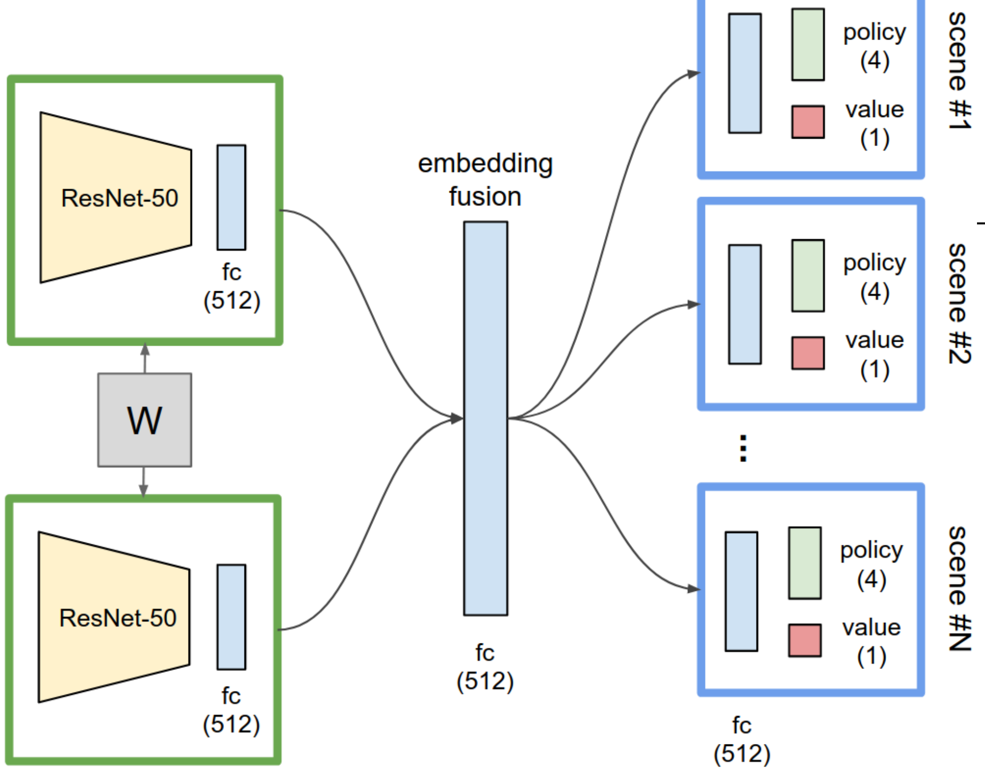
1. -0.01 for every time step
2. -0.1 for every illegal action like collision
3. 10 if it successfully navigates to the target

The episode ends if either we have exceeded the number of steps available, or we have successfully navigated to the target

```
def get_reward(self):
    if self.terminal:
        return 10
    return -0.01 if not self.action_failure else -0.1
```

Agent Architecture

1. Both target and observation are fed into ResNet-50 which has been pre-trained on the imagenet dataset
2. The two outputs from ResNet are fed into two siamese layers with shared weights
3. The two outputs are merged in a shared embedding layer
4. This embedding is passed through scene specific layers to capture the unique characteristics of each scene



TRAINING

Training Hardware

The network is trained on a system with the following specifications

AMD Ryzen 9 5950X

RTX 3070

32 GB RAM

Training Protocol

The current network is trained similarly to the A3C network with various threads trained across different targets at the same time.

1. Each thread is initialized and trained independently
2. The threads share the parameters, which are then combined in the master network

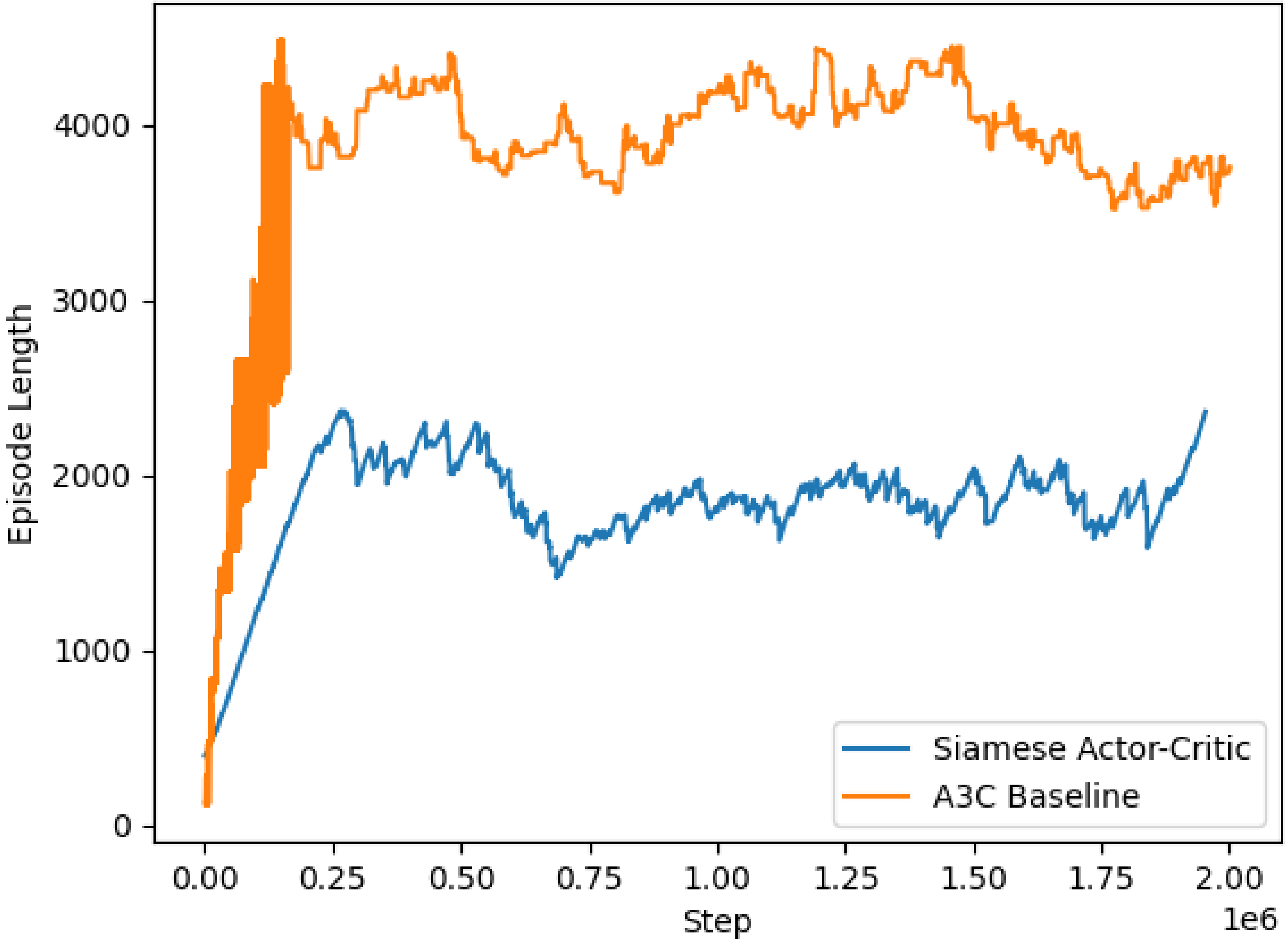
Training across different targets at the same time should help with generalization

Training Parameters

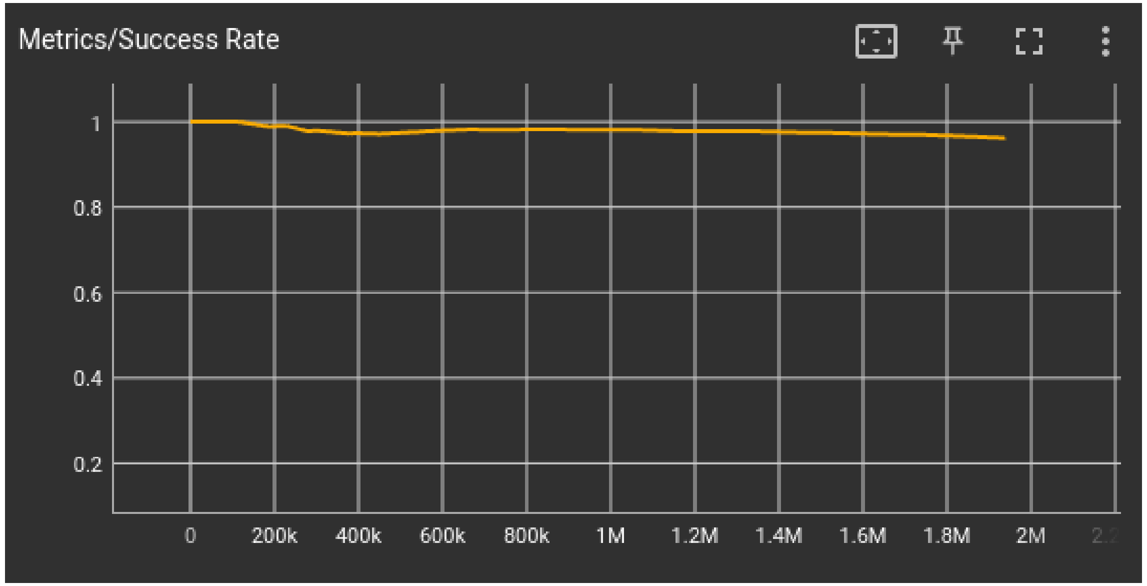
1. Our network
    a. The network is trained across 10 threads for 10 targets across 5 scenes
    b. A shared RMS optimizer is used to train the threads

2. A3C Baseline
    a. The network is trained across 10 threads for 10 targets across 5 scenes

Training Results



Success Rate during training



Our model has a near 100% success rate

Training Demo

Target

Training Demo

Target

Metrics Collected

We are collecting the following metrics to evaluate our model

1. Average Episode Length: The lower the episode length, the better the model

Average Length of Episodes compared to other baselines

Method	Avg episode length
Random Walk	3164
Shortest Path	13.3
A3C	4162
Siamese Actor-Critic	2263

Tested on the training data.

Performance of siamese actor critic is twice as much as compared to A3C, but it is a far cry from the best possible case

Conclusions

1. Our algorithm seems to have fared better than state of the art A3C methods when it comes to generalization across multiple scenes and targets.
2. However, the performance was not much better than a Random walk and far below the best case scenario