IMITATION LEARNING FROM DEMONSTRATION

ABSTRACT. This project addresses imitation learning from video demostrations. Expert videos are collected and feeded as a teacher to network. The general archtecture has two parts: the encoder network and the reinforcement learning network. In the encoder network, a simple network is trained by a loss function based on norms of the frames of the running cartpole and expert vodeos. In the reinforcement learning part, the reward function and loss function are also based on similarity measures computed over the features of the expert video frames and running catpole frames.

INTRODUCTION

This project is based on the research is conducted by Abbeel and Levine et.al. [1] in which the authors presented a context translation archtecture that can learn from videos taken from different context (see Figure 1).

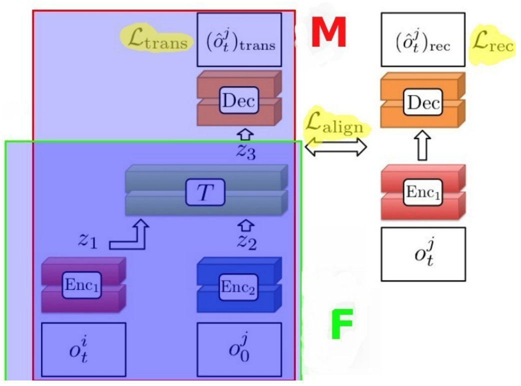
$\mathcal{L}_{\text{trans}}$
$(\hat{o}_t^j)_{\text{trans}}$
M
$(\hat{o}_t^j)_{\text{rec}}$
$\mathcal{L}_{\text{rec}}$
Dec
$\mathcal{L}_{\text{align}}$
Dec
z_3
T
Enc$_1$
z_1
z_2
o_t^j
Enc$_1$
Enc$_2$
F
o_t^i
o_0^j

FIGURE 1. The feature learning architecture in the reference paper

In the standard imitation learning methods assume that the agent receives examples of observation-action tuples that could be provided, for instance, to a supervised learning algorithm (cloning version of imitation learning). The idea behind is to observe another person performing some behavior and then figure out which actions will realize that behavior, also realizing for changes in viewpoint, surroundings, and embodiment. For Reinforcement Learning (RL) they used policy gradient with TRPO which is provided in `rllrab`.

Key words and phrases. imitation learning, machine learning, reinfocement learnign, CartPole problem.

Deep Reinforcement Learning Course Project.

1. The Architecture

The architecture of our imitation learning algorithm is also consists of two parts: the learning reward function part and RL part. Figure 2 represents the complete architecture that we configures. The architecture consists of three main parts: pre-

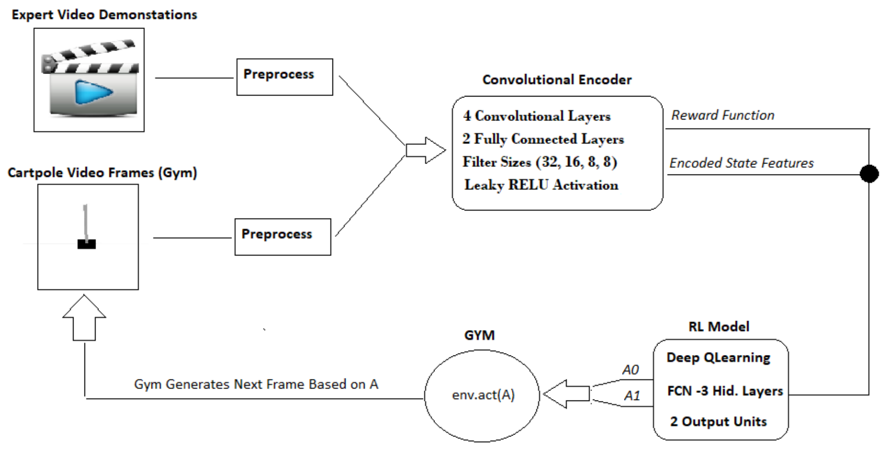

Figure 2. The general architecture that we used in the project

processing the videos, learning the reward (Encoder part) and the RL part.

We were using the openai gym environment for generating videos. There are many environments in the openai-gym to experiment in the project, I selected the CartPole (shown in the Figure 3). In the gym CartPole version [2], the state is

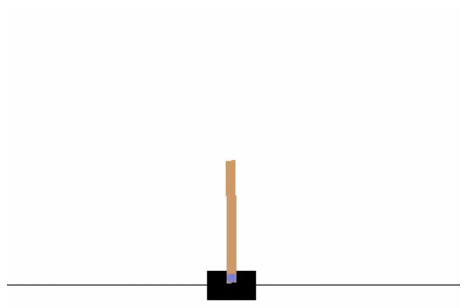

Figure 3

a vector with size 4 in which is described in the Table 1. There are two options: Push cart to the left (0) and Push cart to the right (1). Reward is 1 for every step taken, including the termination step Starting State. All observations are assigned a uniform random value between ±0.05.
We first trained a Q-Learning agent that performs reasonably well in the Cartpole

TABLE 1. The states of gym version of the CartPole

Num	Observation	Min	Max
0	Cart Position	-2.4	2.4
1	Cart Velocity	$-inf$	inf
2	Pole Angle	-41.8	41.8
3	Pole Velocity at Tipe	$-inf$	inf

problem, and generated videos from this agent that we considered as our expert. However, there were variation in these experts, as some were focused mostly in the middle region, and others were shifting to the left or right, while still able to balance the pole itself. Figure 4 below shows the differences in the tracks explored by some agents.

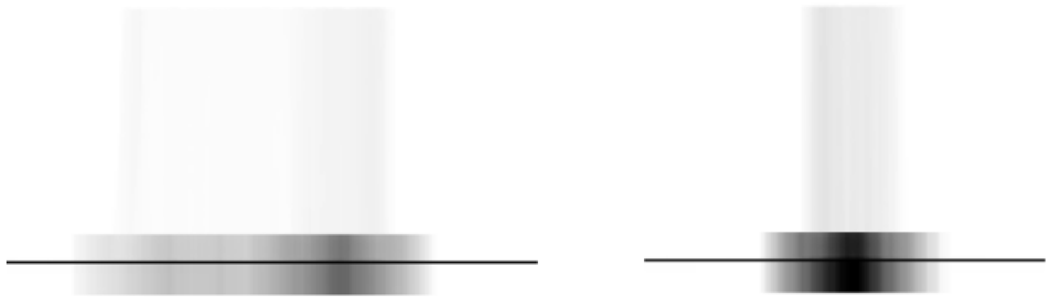

FIGURE 4. Area explored by different expert agents

1.1. **Preprocessing.**

1.1.1. *Feature Tracking.* The first component of the project involved extracting meaningful features from the video frames and tracking them through time. Given that the original dimensions of the images were 400×600, we had to resize them to a lower resolution of 36 x 64 in order to speed up the training process and feed them to the convolutional encoder to compute the features. However, performing this operation meant that we would lose a lot of information and therefore we did not really get meaningful results by performing this operation alone. Therefore, we decided to get some information from the high resolution images that we would use in conjunction with with the low resolution features from the convolutional encoder, and learn a weighted reward function by using these 2 approaches.
It is important to note the key characteristics we wanted our feature extractor to have.

(1) Same number of features extracted across all images
(2) Same features points extracted across all images
(3) Meaningful features extracted

We will now explore the different methods we tried to get the features from the high resolution images.

SIFT. The first approach we tried in order to extract features from the image was Shift Invariant Feature Transform. While this method provided us with good feature descriptors that met condition (3) and partly condition (2) above, it had some trouble meeting condition (1). The reasons for this phenomena are beyond the scope of this report, but they are illustrated in Figure 5 below: In the figure

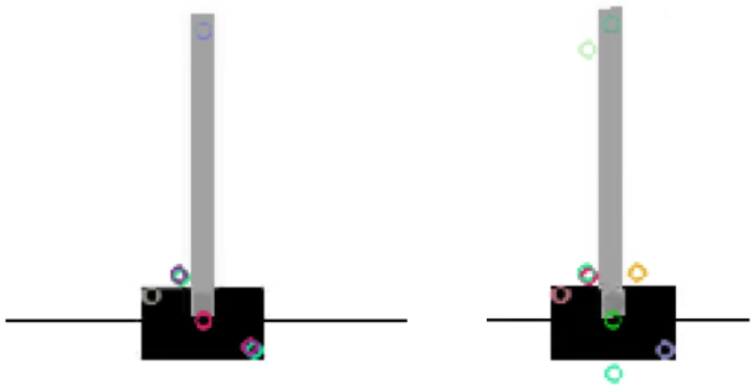

FIGURE 5. SIFT Features learnt on 2 different images

above, we were extracting the same number of SIFT features but they gave different outputs (with some noise as well), and given that we need to extract the same features, this proved to be quite problematic. Of course we could have performed feature matching to get the same features in both images, which is what we would have done, but we found a better algorithm to apply.
Good Features For Tracking: The second algorithm we tried out in getting the features from the image was the good features to track algorithm. The good thing about this algorithm is that it met all conditions (1), (2) and (3) above and as such we decided to use it for tracking. Figure 6 below shows the features generated by this algorithm. Since this algorithm gave us the same features in most of the images generated from the environment, we could now track the optical flow of these features, combine these with the reward function we learnt from the convolutional encoder and get a weighted reward. While the results were not that great, they were still somewhat reasonable, and there was evidence of learning from both the learning curves as well as visually. If you wait long enough, you can see that the agent starts learning how to swing in order to survive longer.

1.2. **Reinforcement Learning.** The reinforcement learning (RL) networks is in the second part of the architecture in which we are feeding the learned rewards to the RL and asking to learn the CartPole optimal policy.
In this porject we used different RL methods Q-learning and Policy Gradient methods;

1.2.1. *Q-learning.* We tried Q-learning method with different parameters and different structures, The results in general were not satisfactory. Deep Q-learning is applied as a RL algorithm. In this case the reward is obtained from the learning features process. The Deep Q-learning code is provided in lms.
In the deep Q-learning case we used single network for learning the reward which

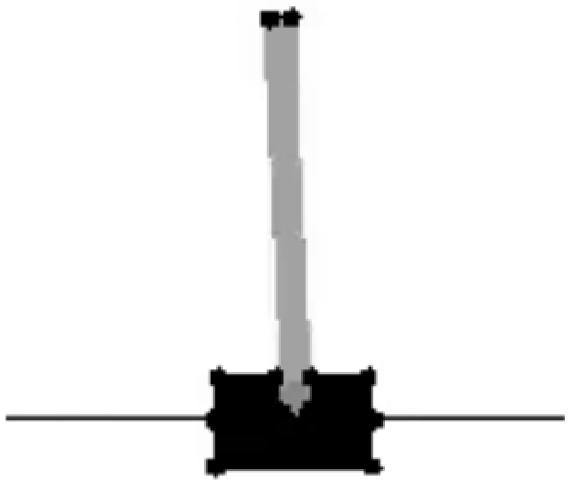

FIGURE 6. GFTT Features on Cartpole Image

was mainly based on the papers reward learning approach. This architecture uses low resolution image features in order to extract image features.
There are two fully connected hidden layers with 50, 50 arrangement. Reward in this case is computed with the following formula:

$$R = \|F^{lr}_{exprt} - F^{lr}_{agent}\|_2$$

here F^{lr}_{exprt} is the expert video frame features from high resolution images and F^{lr}_{agent} is the same for agent.

1.2.2. *Policy Gradient.* The second RL algorithm that we applied in this project is REINFORCE Monte Carlo Policy Gradient AI Player which is provided in lms with different hyperparametes. Network and cost function is designed with tensorflow which seems to be easier than other platforms in designing cost function and preprocessing.
We tried many different structures of the network, shallow - to - deep networks to see if there is some significant effects.

Reward Function

Designing an encoder in order to learn reward function requires subtlety so that it can learn proper features. The reward function learning procedure that we designed in our project is different from what is proposed in the paper because our setup was different and we were also ignoring context translation in the project.
To find the ideal weights in using the high dimensional and low dimensional features combination, we manually tried a different set of weights and tried to find the one that would gives us the best results (perhaps we could also define a network to do this). Note that the first reward function was generated from Inverse Reinforcement Learning (IRL) network that takes the low resolution input as an image, creates

a feature representation based on learned weights from the expert, and gets the reward function as the root of the norm between the encoded features of the expert and that of the agent:

$$R_1 = \|F^{hr}_{exprt} - F^{hr}_{agent}\|_2$$

where F^{hr}_{exprt} is the expert video frame features from high resolution images and F^{hr}_{agent} is the same for agent.

$$R_2 = \|F^{lr}_{exprt} - F^{lr}_{agent}\|_2$$

here F^{lr}_{exprt} is the expert video frame features from high resolution images and F^{lr}_{agent} is the same for agent. The second reward function was learnt from the high resolution image features, however since the differences between these features were much smaller, we tried to emphasize these differences when designing the reward function. Based on these two the RL reward is computed by the following formula:

$$R = w_1 R_1 + w_2 R_2$$

where $w_1 = 0.1$ and $w_2 = 0.1$. This reward function is tested in different experiments with CartPole env and we observed that it workes better that other options however those may not be optimal. Designing a new network to find these two parameters is easy, but the problems is that considering the computation vs what we get as a results it does not worth to set another network here, so we preferred to use the parameters that we obtained by experience.

2. The Results

The networks are trained with 82 videos that we taken from trained CartPole agent at gym environment (however bacause of the size issue `10GB` we upload just 5 videos).
The training is done with different parameters and network designs. The best result that we got until now is shown in the following Figure 7. This result is produced by `RL_Network_Conv_Policy.py` provided in the code folder.

References

[1] YuXuan Liu, Abhishek Gupta, Pieter Abbeel, Sergey Levine, Imitation from Observation: Learning to Imitate Behaviors from Raw Video via Context Translation, Submitted to NIPS 2017, Long Beach. YuXuan Liu and Abhishek Gupta had equal contribution. arXiv:1707.03374 [cs.LG]

[2] `https://github.com/openai/gym/wiki/CartPole-v0`

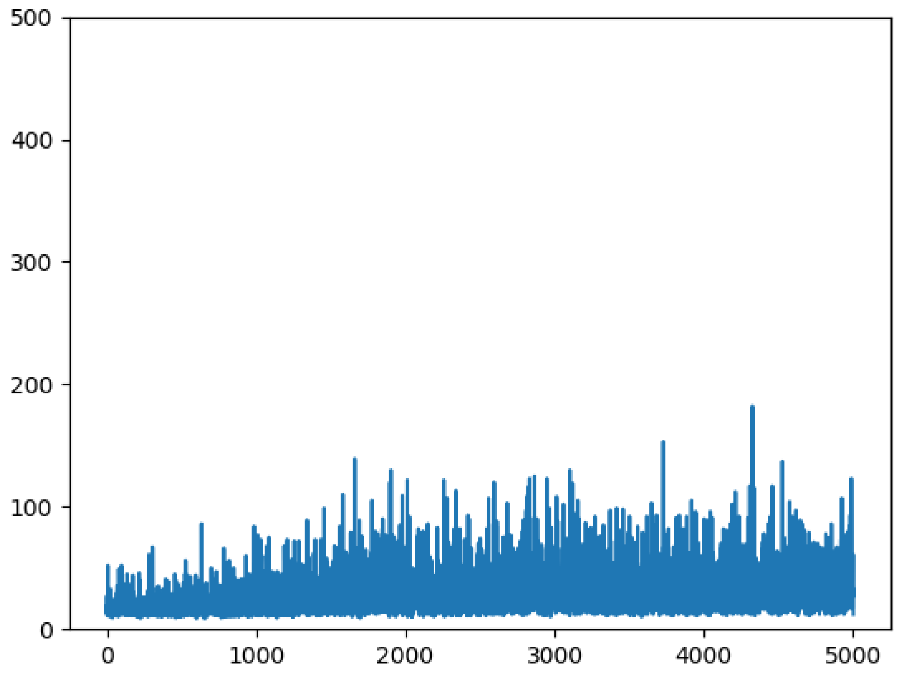

Figure 7. The results obtained from the policy learning RL after same iterations as Q learning. It is learning with small learning rate $lr = 0.0001$ with AdamOptimizer in tensorflow.