YODO: You Only Distil Once

Aditya Kompella
(apk74)

Adrian Hilton
(mah528)

Thomas Patton
(tjp93)

1 Introduction

Model-based approaches have long dominated model-free approaches in literature and in practice. Learning a world model and then planning with that world model using a technique like MPC is state of the art for control tasks in many cases. While this approach can be good, planning into the future using MPC can computationally and temporally expensive and the policy is quite slow compared to some light-weight model-free rl approaches that train a policy network.

In this paper, we present the novel YODO (You Only Distil Once) algorithm which uses DAgger to distill expert information from an MPC into networks of three varying sizes. We then implement our YODO algorithm in the `parking-v0` environment to illustrate that these distilled models are significantly faster than our MPC while only having marginally less performance.

2 Problem

2.1 Introduction

As mentioned above, model-based learning approaches have been massively influential in the last few years. However, a planning method like MPC is often incredibly slow: in the envioronment we work in in this paper (`parking-v0`) episode generation takes an average of a second per generated state which is far too slow for real-world inference. By contrast, model-free approaches have incredibly quick inference times but can often struggle to effectively learn more complicated environments on their own.

Our work in this paper effectively combines the benefits from both of these approaches. By using DAgger with our MPC treated as the expert, we are able to effectively distil this knowledge into relatively simple networks while still being able to retain a large fraction of the performance.

2.2 Related Work

In *[1]* the authors tackle a similar problem by distilling the policy of an agent that performs at the expert level into a new, smaller network that performs at the same level while being orders of magnitude more efficient. However the authors of ths paper use a DQN for distillation rather than an MPC.

In *[2]* the authors also attempt to solve a similar problem but is orders of magnitude more complex, using a mixture-of-experts (MEN) network and divides each expert over a discrete element of a real robot. Instead, our focus is on using simple imitation learning with DAgger to solve a much more contained environment.

2.3 Implications

The success of this idea would present a much easier way to train policy neural networks that outperform other model-free methods.This could lead to more efficient and effective training of policy neural networks and improved performance in a wide range of applications.

3 Approach

3.1 Environment

For our approach, we used the `parking-v0` environment from the `highway-env` collection of environments in Gymnasium. At a high level, this environment is a two-dimensional top down view of a car in a parking lot where the goal is to park the car into the specified space. In a general sense, the agent is rewarded based on the weighted L1 norm of the (x, y, θ) position of the car and the (x', y', θ') position of the goal.

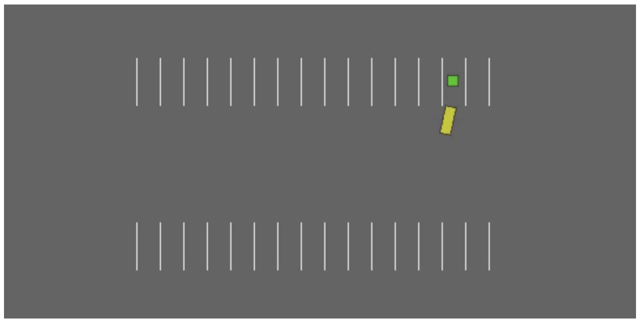


Figure 1: An example of the environment in Gymnasium

3.2 Distillation Process

In order to create our distilled model, we performed the following steps:

- First, we load in the `parking-v0` environment and use it to sample 1000 episodes each with a maximum length of 100 timesteps.
- We then create a dynamics model for our MPC by creating a small neural network with fully connected layers and ReLU activation functions. We train this model with 1500 epochs of gradient descent to predict the next state s_{t+1} given a state-action pair (s_t, a_t).
- After this, we use a Cross-Entropy Method (CEM) algorithm to find optimal trajectories of actions thus allowing us to create the planner of our MPC. Fundamentally, this works by sampling trajectories from a probability distribution and updating the distribution by using the top-k trajectories that minimize the cross-entropy to update the distribution.
- We then use DAgger to distil our CEM into three policies of varying size for a more comprehensive evaluation of our methods. These policies are implemented as fully-connected networks as shown in *Figure 4*. For DAgger we used 5 iterations with 100 steps per iteration.
- As a metric for evaluation, we also use actor-critic (A2C) to train an off-the-shelf implementation of `stablebaselines`.
- Finally, we evaluate our three distilled models. For comparison we compare our models against our baseline MPC and the `stablebaselines` approach by evaluating on **computation time per episode** and **mean reward per episode**.

4 Results

As was mentioned previously, the five models to be evaluated are our MPC, an off-the-shelf implementation of `stablebaselines`, and our three distillation models (small, medium, and large).

4.1 Computation Time

For computation time, we generated 100 episodes for each algorithm and recorded the mean time for each episode generation in milliseconds. Here a smaller (i.e. faster) time is better:

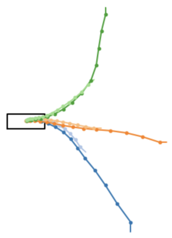


Figure 2: Example trajectories predicted by our dynamics model. The green, orange, and blue lines represent the predicted trajectories for steering angles of $[-0.5, 0, 0.5]$ radians respectively

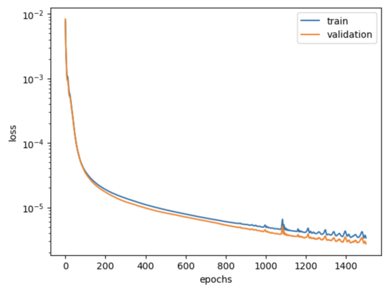


Figure 3: The training and test curves for our dynamics model. Using mean squared error our final train and test errors are on the order of 10^{-5} indicating our models are very accurate.

Model	Execution Time (in ms)
`MPC`	160
`distil_tiny`	**0.633**
`distil_medium`	2.51
`distil_large`	4.93
`A2C`	0.754

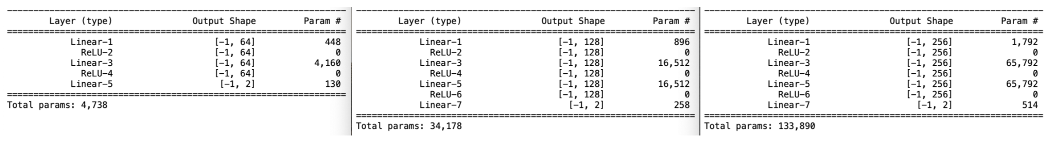
4.2 Mean Reward

In the parking environment the reward is the negative L1 norm between the car's (x, y, θ) position and the (x', y', θ') position of the parking space, so closer to zero is better.

Model	Mean Reward
`MPC`	**-0.17**
`distil_tiny`	-0.256
`distil_medium`	-0.212
`distil_large`	-0.190
`A2C`	-0.967

4.3 Analysis

Likely the most staggering result here is the abysmal inference time for the MPC compared to the other methods. Using a model-free approach is faster by several orders of magnitude. By contrast, we only see a minor degradation in performance in comparing the rewards. In our `parking-v0` environment all of the above methods (besides the actor critic) had great convergence. In general this confirms our hypothesis: all three of our distilled models achieve performance comparable to the MPC baseline while keeping the speedy inference of a model-free approach. Although model-based methods achieve the highest accuracy, the overhead of running a planner makes our distilled approach better for real-time robotics implementations.



```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Linear-1                   [-1, 64]             448
              ReLU-2                   [-1, 64]               0
            Linear-3                   [-1, 64]           4,160
              ReLU-4                   [-1, 64]               0
            Linear-5                    [-1, 2]             130
================================================================
Total params: 4,738
```

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Linear-1                  [-1, 128]             896
              ReLU-2                  [-1, 128]               0
            Linear-3                  [-1, 128]          16,512
              ReLU-4                  [-1, 128]               0
            Linear-5                  [-1, 128]          16,512
              ReLU-6                  [-1, 128]               0
            Linear-7                    [-1, 2]             258
================================================================
Total params: 34,178
```

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Linear-1                  [-1, 256]           1,792
              ReLU-2                  [-1, 256]               0
            Linear-3                  [-1, 256]          65,792
              ReLU-4                  [-1, 256]               0
            Linear-5                  [-1, 256]          65,792
              ReLU-6                  [-1, 256]               0
            Linear-7                    [-1, 2]             514
================================================================
Total params: 133,890
```

Figure 4: The network architectures of the three different model sizes we distilled into via the MPC.

5 Contributions

- **Aditya Kompella** was responsible for implementing the MPC for the Gym environments and using it to create training data for the distillation model.
- **Adrian Hilton** was responsible for everything related to the implementation and execution actor-critic to refine `stablebaselines` .
- **Thomas Patton** was responsible for implementing the DAGGER algorithm to actually distil the MPC into the smaller network.

All three group members contributed equally to evaluating the models, writing the paper, and creating the final video.

References

[1] Andrei A. Rusu, Sergio Gomez Colmenarejo, Caglar Gulcehre, Guillaume Desjardins, James Kirkpatrick, Razvan Pascanu, Volodymyr Mnih, Koray Kavukcuoglu and Raia Hadsell. Policy Distillation, 2015; arXiv:1511.06295.

[2] Alexander Reske, Jan Carius, Yuntao Ma, Farbod Farshidian and Marco Hutter. Imitation Learning from MPC for Quadrupedal Multi-Gait Control, 2021; arXiv:2103.14331. DOI: 10.1109/ICRA48506.2021.9561444.