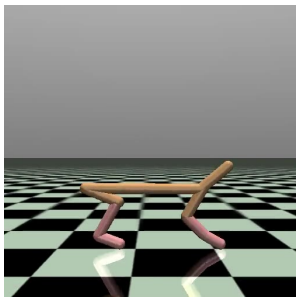
# LaDy (Latent Dynamics) Duckie - Model Based RL in Gym-Duckietown

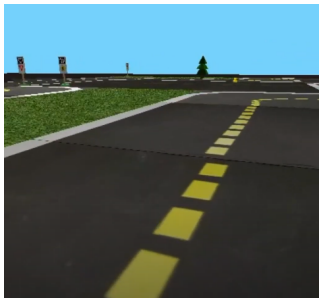
**Alihusein Kuwajerwala** [*1,2] **Maxime Fournier** [*1] **Brahim Ben Malek** [*1]

## Abstract

Model-based reinforcement learning (MBRL) algorithms have many complex sub-components that each need to be carefully selected and tuned, which may explain why MBRL approaches often struggle to match the performance of model-free methods on various tasks. In this work, we aim to make it easier to evaluate and improve MBRL approaches by integrating the Gym-Duckietown environment (Paull et al., 2017) into MBRL-Lib (Pineda et al., 2021), an open source library of popular MBRL algorithms. Our goal is to enable MBRL approaches to be easily deployed in the Duckietown environment without significant software overhead. Additionally, we apply the MBRL approach from 'PlaNet' (D. Hafner & Davidson, 2019), attempting to learn a dynamics model for the Duckietown simulation, and compare its performance on the Duckietown and 'Cheetah Run' environments. The code can be found at: `https://github.com/alik-git/mbrl-lib`



(a) HalfCheetah-v2



(b) Gym-Duckietown

(c) Overhead view of the Gym-Duckietown environment

*Figure 1.* Images of the environments we used for this work. For each, we use the PlaNet algorithm to learn a dynamics model and train a reinforcement learning agent.

## 1. Introduction

The format of this paper is as follows:

- In Sec. 1 we briefly cover the relevant background / related works for this project, namely PlaNet, MBRL-Lib and Gym-Duckietown.
- In Sec. 2 we cover our code that integrates Gym-Duckietown with MBRL-Lib and its functionality.
- In Sec. 3 we discuss the challenges we encountered.
- In Sec. 4 we showcase some experiments using PlaNet approach in Gym-Duckietown and our results.
- Finally, in Sec. 5 and Sec. 6 we reflect on the project as a whole and discuss the potential for future work in this area.

### 1.1. Model Based vs Model Free RL

Reinforcement learning is an active field of research in autonomous vehicles. Model-based approaches are used less frequently than model-free approaches since model-free approaches have had better performance in the past and as a result it is easier to find existing implementations.

The largest advantage that model based approaches offer is their superior sample complexity. That is, model based approaches can use orders of magnitude less data or interaction with the external environment compared to model-free methods. This is because model based methods can train the policy on the internal model of the environment as opposed to the external environment itself. Also, model-free methods implicitly learn a 'model of the environment' in some way eventually in order to predict the long-term reward of an action, they just do so inefficiently. Additionally, another advantage that model-based methods have is that the learned model of the environment can be task agnostic, meaning

[*]Equal contribution [1]Université de Montréal, Montréal, Canada [2]Mila - Quebec AI Institute, Montreal, Canada. Correspondence to: Alihusein Kuwajerwala <alihusein.kuwajerwala@umontreal.ca>.

*IFT 6269 project report - Université de Montréal*

that it can be used for any task that requires predicting the state of the environment in the future.

### 1.2. MBRL-Lib

There are many available open source implementations of popular model-free approaches (Dhariwal et al., 2017; Kostrikov, 2018) and model-based approaches (Dong et al., 2019). During our review, we found the MBRL-Lib toolbox (Pineda et al., 2021) to be most useful.

MBRL-Lib contains implementations of various popular model-based approaches, but more importantly is designed to be modular and compatible with a wide array of environments; and makes heavy use of configuration files to minimize the amount of new code needed to tweak an existing approach. Integrating the Gym-Duckietown environment to work with this toolkit would allow users to easily experiment with and evaluate a variety of model based RL approaches in the Duckietown simulation.

### 1.3. Gym-Duckietown

The environments in which to deploy RL methods can be as simple as tic-tac-toe or as complex as a nearly photo realistic simulation of a whole city (Dosovitskiy et al., 2017). The most popular environments for current RL research are those provided in the OpenAI Gym toolkit (Brockman et al., 2016). OpenAI Gym also provides a set of standards which can be used to make environments widely compatible with any software designed to accommodate those standards. MBRL-Lib is an example of such software, and comes with the standard environments from OpenAI Gym built in.

Naturally, this makes Gym-Duckietown—a self-driving car simulator for the Duckietown universe already built as an OpenAI gym environment (Paull et al., 2017)—the perfect candidate to make available for use with the model-based approaches provided by MBRL.

This is not to say however, that Gym-Duckietown works perfectly with MBRL-Lib right out of the box. Duckietown has significantly more complex dynamics than the standard OpenAI Gym environments. Consider for example that the Cheetah environment (See Fig. 1a) only consists of one (albeit complex) object in a plain background with the camera always tracking it. Compared to Gym-Duckietown, where the camera is fixed on the car which moves through the scene, drastically changing the objects found in different observations. Our results indicate that while MBRL methods have the potential to perform well in Gym-Duckietown, they need to be carefully tuned and modified to achieve results comparable to those of the current baselines.

### 1.4. PlaNet

The reason model-based approaches have often struggled to match the performance of model-free approaches despite their superior sample complexity is that it is difficult to accurately learn a dynamics of a large and complex environment. The effects of lighting, contact dynamics, non-linear motion etc. are nearly impossible to account for explicitly in enough detail for large scenes. This has prevented model-based approaches from accurately predicting states many steps into the future.

Recent advances in model-based approaches have made them more competitive by leveraging a latent space representation of the world. By using an encoder network to map the environment's high dimensional observations into low dimensional embeddings and learning the dynamics model in the latent space, approaches like PlaNet (D. Hafner & Davidson, 2019) can be used to learn a dynamics model capable of accurate predictions for a complicated environment. This also has significant computation efficiency gains, as it is much faster to propagate dynamics forward in the latent space as opposed to in an explicit simulation. The core of PlaNet's approach also involves both deterministic and stochastic transition components and a multi-step variational inference objective that they call 'latent overshooting'.

In the original paper, the authors evaluate PlaNet on environments in the DeepMind control suite (Tassa et al., 2018), but these environments are also available as part of OpenAI Gym, which is what MBRL-Lib and this project use for evaluation.

## 2. Integrating Gym-Duckietown & MBRL-Lib

### 2.1. Changes to MBRL-Lib

As noted before, the MBRL-Lib code is well organized, and adding the Gym-Duckietown environment as a configurable option was straightforward. We've included the code here:

```
# Use the duckietown environment with
# settings that closely match the settings
# used for cheetah run
if cfg.overrides.env=="duckietown_gym_env":
    env = mbrl.env.DuckietownEnv(
        domain_rand=False,
        camera_width=64,
        camera_height=64
    )
    term_fn = \
    mbrl.env.termination_fns.no_termination
    reward_fn = None
```

*Listing 1.* Adding Gym-Duckietown to MBRL-Lib

The main pre-requisite to being able to do this is to import the Gym-Duckietown code as an external python module into MBRL-Lib so that the `DuckietownEnv` object can be

used directly without creating any new classes or objects. We took efforts throughout this project to avoid re-writing any code already found in either code base, and to import and use the original code directly.

Next we add a configuration file for default parameter values associated with the Duckietown environment. This is a YAML file called `planet_duckietown.yaml` and for now only contains default values for the PlaNet algorithm, however more config files can be easily created for other algorithms as long as the follow the same format.

We ran into some issues with using the observations from `DuckietownEnv` with the PlaNet algorithm, we expand on this in the Challenges section.

### 2.2. Changes to Gym-Duckietown

Other than a couple of minor bug fixes and tweaks we did not need to make any changes to Gym-Duckietown to use it with MBRL-Lib . We turn off domain randomization and change the default observation size (in `[height, width]`) to be `[64,64]` as opposed to `[640, 480]` as we wanted to closely match the hyper-parameters used for the PlaNet on the HalfCheetah-v2 environment as a starting point. Another practical issue is simply that using observations of size `[640,480]` maybe require a very large amount of video memory (up to 3 TB) depending on the chosen batch size. To be clear, the choice of size `[64,64]` is arbitrary and is subject to future experimentation.

### 2.3. Integrating W&B

MBRL-Lib uses Hydra (Yadan, 2019) for logging and organizing runs. They also have some basic visualization tools specific to MBRL techniques that they have built into their library. These tools however still lack the ability to automatically plot metrics for each run, and back up metrics off-site, filtering runs by a certain metric, and it can be quite cumbersome to wade through hundreds of folders in the Hydra logging directory to find and interpret your results.

To alleviate this issue, we import and use the Weights & Biases library (Biewald, 2020) to track our runs. We also log the configuration file, and log a copy of every metric logged by Hydra in MBRL-Lib. This gives an online dashboard that automatically organizes our runs and plots their metrics, as well as all the functions we mentioned earlier.

## 3. Challenges

### 3.1. Observation shape mismatch

The most challenging issue to make Gym-Duckietown work with MBRL-Lib was that the PlaNet encoder model expects the observations to be of the following shape: `[batch_size, color_channels, image_width , image_height]`. However the `DuckietownEnv` returns observations in the shape `[image_width, image_height, color_channels]`. The problem here is that the PlaNet encoder expects `color_channels` to be before `image_width` and `image_height`. We found no way to adjust the the model to accomodate this shape via the config files, nor a way for Gym-Duckietown to return observations with the dimensions permuted. As a result, we were forced to manually permute the observation Tensor before it is fed to the PlaNet encoder , and permute it back again after the resulting embedding is fed through the PlaNet decoder. These modifications were made in the `planet.py` file in the `mbrl/models` folder.

A better way to do this would be to have pre-processing function for the Gym-Duckietown observations, but since the MBRL-Lib code is designed to be quite modular, we could not find a clean way to add such a function without disturbing some other part of the code.

### 3.2. Importing Gym-Duckietown

Since Gym-Duckietown itself is not a pip package one can import from PyPi, we have included instructions for how to manually import it in the `README.md` file for our project.

### 3.3. Hyperparameter Tuning

As noted earlier, MBRL methods are quite sensitive to the choices of hyperparameters and the environments we are using are complex, so training times can be significant. None of the configurations we tried achieved consistently positive rewards, even when left to train overnight. Therefore, for the scope of this project we chose to use the same hyperparamters that the PlaNet authors had success with the HalfCheetah-v2 environment, which we consider to be our baseline for comparison.

In our results (See Fig. 2, 3), we observe a positive trend in both the loss curve and the reward curve overtime for the Gym-Duckietown environment, although as expected the overall results are much better for the HalfCheetah-v2 environment, which the parameters are designed to suit. These results can be treated as a starting point for model based reinforcement learning on the Gym-Duckietown environment.

### 3.4. Other Misc Minor Issues

There were a handful of other minor issues we had to solve for the project. These included:

- Commenting out / updating some deprecated code in Gym-Duckietown.
- Fixing an off by one error in the sequence transition

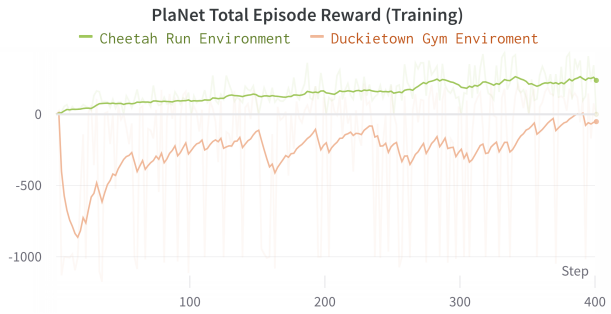



*Figure 2.* A comparison of PlaNet's Total Episode Reward on the Gym-Duckietown and HalfCheetah-v2 environment on the 'Cheetah Run' task.

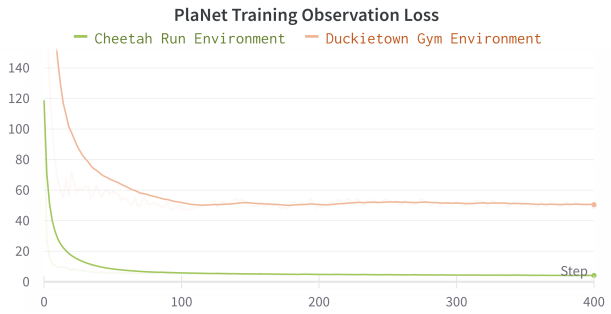



*Figure 3.* A comparison of PlaNet's Training Observation Loss on the Gym-Duckietown and HalfCheetah-v2 environment on the 'Cheetah Run' task.

sampler in MBRL-Lib that would try to sample a transition even when the current batch was empty.

- Creating a copy of the configuration object that was not nested, because that would prevent W&B from being able to parse it.

## 4. Experiments

As noted in Sec. 3.3, we used the same PlaNet hyperparameters for Gym-Duckietown that the PlaNet authors chose for the 'Cheetah Run' task on the HalfCheetah-v2 environment. Our results are shown in Fig. 2 and Fig. 3 where we observe that while the PlaNet model performs significantly better on the HalfCheetah-v2 environment, it still improves overtime on the Gym-Duckietown environment.

Each of the runs shown took approximately 10 hours on an NVIDIA RTX 2070 GPU. In the HalfCheetah-v2 environment the PlaNet model can achieve positive rewards from the beginning whereas for the Gym-Duckietown environment it takes a significant amount of time for the average reward to be positive. We did not have time within the scope of this project to have runs long enough to consistently observe positive rewards on the Gym-Duckietown environment.

However the trend we do observe does indicate that given enough time and compute resources, the PlaNet model could learn a useful dynamics model and policy for the Gym-Duckietown environment. This may also be explained by the fact that the Gym-Duckietown environment is much larger, and has a larger collections of objects than the HalfCheetah-v2 environment. Also the two environments are fundamentally different in that the camera observes the vehicle's point of view and moves freely around the science in Gym-Duckietown as opposed to HalfCheetah-v2 where the camera always tracks the cheetah.

## 5. Discussion

Our hope for this project is that it may be used as a starting point for more experimentation with MBRL methods in the Gym-Duckietown environment. Our results indicate that there is certainly potential for significant performance gains by (1) more compute resources, and (2) hyperparameter tuning for PlaNet to better suit the Gym-Duckietown environment. There is also the option of trying different approaches all-together other than PlaNet such as PETS (Chua et al., 2018) and MPBO (Janner et al., 2019), these are already implemented in MBRL-Lib and therefore are great candidates for experimentation.

## 6. Conclusion

In this work, we integrate the Gym-Duckietown environment into the MBRL-Lib toolbox to enable easy experimentation of MBRL approaches for the autonomous driving task within the Duckietown simulator. Our code is open source and well documented, and we provide thorough instructions on how to setup and run it. We also showcase some preliminary experiments applying the PlaNet approach to the Gym-Duckietown environment with results that indicate the potential for future MBRL research to achieve significant performance improvements on the autonomous driving task.

## References

Biewald, L. Experiment tracking with weights and biases, 2020. URL https://www.wandb.com/. Software available from wandb.com.

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., and Zaremba, W. Openai gym. *CoRR*, abs/1606.01540, 2016. URL http://arxiv.org/abs/1606.01540.

Chua, K., Calandra, R., McAllister, R., and Levine, S. Deep reinforcement learning in a handful of trials using proba-

bilistic dynamics models. *CoRR*, abs/1805.12114, 2018. URL http://arxiv.org/abs/1805.12114.

D. Hafner, T. Lillicrap, I. F. R. V. D. H. H. L. and Davidson, J. Learning latent dynamics for planning from pixels. arXiv:1811.04551v5, 2019. URL https://arxiv.org/abs/1811.04551v5.

Dhariwal, P., Hesse, C., Klimov, O., Nichol, A., Plappert, M., Radford, A., Schulman, J., Sidor, S., Wu, Y., and Zhokhov, P. Openai baselines. https://github.com/openai/baselines, 2017.

Dong, L., Gao, G., Li, Y., and Wen, Y. Baconian: A unified opensource framework for model-based reinforcement learning. *CoRR*, abs/1904.10762, 2019. URL http://arxiv.org/abs/1904.10762.

Dosovitskiy, A., Ros, G., Codevilla, F., Lopez, A., and Koltun, V. CARLA: An open urban driving simulator. In *Proceedings of the 1st Annual Conference on Robot Learning*, pp. 1–16, 2017.

Janner, M., Fu, J., Zhang, M., and Levine, S. When to trust your model: Model-based policy optimization. *CoRR*, abs/1906.08253, 2019. URL http://arxiv.org/abs/1906.08253.

Kostrikov, I. Pytorch implementations of reinforcement learning algorithms. https://github.com/ikostrikov/pytorch-a2c-ppo-acktr-gail, 2018.

Paull, L., Tani, J., Ahn, H., Alonso-Mora, J., Carlone, L., Cap, M., Chen, Y. F., Choi, C., Dusek, J., Fang, Y., Hoehener, D., Liu, S.-Y., Novitzky, M., Okuyama, I. F., Pazis, J., Rosman, G., Varricchio, V., Wang, H.-C., Yershov, D., Zhao, H., Benjamin, M., Carr, C., Zuber, M., Karaman, S., Frazzoli, E., Del Vecchio, D., Rus, D., How, J., Leonard, J., and Censi, A. Duckietown: An open, inexpensive and flexible platform for autonomy education and research. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1497–1504, 2017. doi: 10.1109/ICRA.2017.7989179.

Pineda, L., Amos, B., Zhang, A., Lambert, N. O., and Calandra, R. Mbrl-lib: A modular library for model-based reinforcement learning. *Arxiv*, 2021. URL https://arxiv.org/abs/2104.10159.

Tassa, Y., Doron, Y., Muldal, A., Erez, T., Li, Y., de Las Casas, D., Budden, D., Abdolmaleki, A., Merel, J., Lefrancq, A., Lillicrap, T., and Riedmiller, M. Deepmind control suite, 2018.

Yadan, O. Hydra - a framework for elegantly configuring complex applications. Github, 2019. URL https://github.com/facebookresearch/hydra.