Comparison of Deep Reinforcement Learning and Model Predictive Control for Adaptive Cruise Control

Yuan Lin, Member, IEEE, John McPhee, Member, IEEE, and Nasser L. Azad, Member, IEEE

Abstract—This work compares Deep Reinforcement Learning (DRL) and Model Predictive Control (MPC) for optimal control. We consider a longitudinal Adaptive Cruise Control (ACC), i.e., car-following control, problem whose state-space equations have three state variables. Within the state-space representation, a first-order system is used as the Control-Oriented Model (COM) to describe the acceleration dynamics of a 2015 Toyota Prius hybrid electric vehicle. The multi-objective cost function for car following penalizes gap-keeping errors, control efforts, and jerks. Based on the state-space equations and the cost function, we trained a DRL policy using the Deep Deterministic Policy Gradient (DDPG) algorithm and formulated the MPC optimization problem to be solved via Interior-Point Optimization (IPO). We first tested the DRL-trained policy and the MPC approach with the COM, considering no modeling errors. The COM test results showed that the DRL solution is equivalent to MPC with a sufficiently long prediction horizon with regards to episode cost. Particularly, the DRL episode cost is 4.29% higher than a benchmark solution provided by optimizing the entire episode via IPO. Then the DRL-trained policy and MPC were tested with a High-Fidelity Model (HFM) of the Prius, considering modeling errors. We found that the DRL-trained policy is more tolerant to modeling errors than MPC regarding episode costs. When tested with the Highway Fuel Economy Test (HWFET), the Federal Test Procedure 75 (FTP-75), and the US06 Supplemental Federal Test Procedure on the HFM, the MPC episode costs were 49.62%, 30.20%, and 41.22% higher than those of the DRL solution, respectively.

Index Terms—Deep Reinforcement Learning, Model Predictive Control, Adaptive Cruise Control.

I. INTRODUCTION

Reinforcement learning is a learning-based method for optimal decision making and control [1]. In reinforcement learning, an agent takes an action based on the environment state and consequently receives a reward. Reinforcement learning maximizes cumulative discounted reward by learning an optimal state-action mapping policy through trial and error. The policy is trained via Bellman’s principle of optimality which dictates that the remaining actions constitute an optimal policy with regard to the state resulting from a previous action. Deep reinforcement learning (DRL) which utilizes deep (multi-layer) neural nets as policy representations has drawn significant attention as its trained policy surpassed the best human player in playing board games [2]. Different DRL algorithms have been proposed which include Deep Q-Networks [3], Trust Region Policy Optimization [4], Proximal Policy Optimization [5], and Deep Deterministic Policy Gradient (DDPG) [6]. In this work, we use DDPG. DDPG outputs continuous control actions by training a deterministic policy offline. DDPG is a popular choice for optimal control, especially for a dynamic system that can be modeled with state-space equations.

Model Predictive Control (MPC) is both the states of the art and practice for optimal control [7]. MPC benefits from a sufficiently accurate COM of the plant dynamics. At each time step, a constrained optimization problem is formulated based on the COM to minimize a defined cost function in a predictive time horizon. The optimization problem is solved online and only the first value of the solved control sequence is applied. At the next time step, this predictive control procedure is repeated with updated states. There are various methods to formulate the optimization problem with the state-space equations and the cost function which include direct single shooting, direct multiple shooting, and direct collocation. There are also various online optimization solvers for MPC which include sequential quadratic programming and interior point optimization (IPO) [8]. In this work, we use IPO with direct single shooting. IPO and its family ultimately solve the formulated optimization problem via Newton-Raphson’s method by successively approximating the root of the cost function derivative. The IPO solution is on the interior of the set described by the inequality constraints and close to the true optimal solution.

Since both DRL and MPC can provide optimal control solutions, it is of research interest to understand their advantages and disadvantages. For our comparison, we consider solving an optimal control problem for a dynamic system represented by a system of state-space equations. We do not consider training an end-to-end (such as image-to-control-action) solution using DRL [3]. Before using the example for comparison details, one could understand some known differences between the two. Firstly, MPC demands online optimization that requires relatively powerful computing devices for real-time applications, which raises monetary concerns. For automotive engineering, hardware-in-the-loop simulations are needed to verify the real-time readiness of MPC before real-world deployment [9]. On the other hand, offline-trained DRL solution representations

1Dr. Yuan Lin is a Postdoctoral Fellow in the Systems Design Engineering Department at the University of Waterloo, Ontario, Canada N2L 3G1. y428lin@uwaterloo.ca
2Dr. John McPhee is a Professor and Canada Research Chair in the Systems Design Engineering Department at the University of Waterloo, Ontario, Canada N2L 3G1. mcphee@uwaterloo.ca
3Dr. Nasser L. Azad is an Associate Professor in the Systems Design Engineering Department at the University of Waterloo, Ontario, Canada N2L 3G1. nlashgarianazad@uwaterloo.ca
are neural nets which result in very little computation time during deployment. Secondly, MPC is theory-based while, up to date, DRL control solutions are black-box neural nets and lack theoretical guarantees. In this work, we do not focus on these known differences about the computing requirement and theoretical guarantees for DRL and MPC.

In this work, we focus on the optimality level (minimum episode cost) that DRL and MPC can achieve without and with modeling errors. In order for fair comparison, we use the same COM for DRL to train a policy and for MPC optimization. Most of the parameter settings are the same for both DRL and MPC except that the DRL reward utilizes a discount factor which is absent in the MPC optimization. This is due to the fact that DRL requires a discount factor less than one for convergence while MPC normally does not include the discount factor. We raised a few questions that guided our research:

1. When there are no modeling errors, i.e. testing on the COM, is DRL or MPC better in achieving the minimum cost? We use IPO to optimize for the entire simulation episode once to obtain a benchmark solution, called the IPO solution, for both DRL and MPC. MPC usually obtains better optimality levels with longer prediction horizons. It may be interesting to see the difference between the DRL solution and MPC with difference prediction horizons. The comparison of the DRL, MPC, and IPO solutions could draw insights on training policies via Bellman’s principle of optimality versus optimizing via Newton-Raphson’s method. It would also show the effect of the discount factor on the optimality-seeking of DRL.

2. When there are modeling errors, i.e., testing on the HFM, does DRL or MPC achieve a lower cost? In this work, we consider disturbances to be within the scope of modeling errors since both modeling errors and disturbances are the differences from the state-space model. In our previous work, we showed that modeling errors due to neglecting vehicle dynamics could cause significantly degraded DRL control performance [10]. As both DRL and MPC suffer from performance degradation due to modeling errors [11], [12], this work could show whether DRL or MPC is better at handling modeling errors given that most conditions are the same.

To answer these questions, we develop both DRL and MPC controllers for ACC car-following control. Car following is one of the most common behaviors of road vehicles. ACC is a type of Advanced Driver Assistance Systems that enable intelligent and automated driving [13]. Automated vehicle development has been a popular interest in academia and industry as it could potentially revolutionize transportation. We develop ACC controllers for the hybrid electric vehicle, 2015 Toyota Prius, since we have previously developed a HFM of it in MATLAB/Simulink [14], [9]. The HFM includes control input execution delay (control delay) of 0.2s, power-train modeling, and external resistances including aerodynamic drag and rolling resistance. Road grade resistant force is not considered as we assume flat surfaces. Note that the COM does not include the control delay.

For the rest of the paper, literature review is documented in Section II; ACC problem formulation is in Section III; Methodologies of DRL and MPC are documented in Section IV; Results are shown in Section V; Section VI draws the conclusion.

II. LITERATURE REVIEW

There are multiple papers in the literature that compare reinforcement learning and MPC performances. In [15], the authors compared reinforcement learning and MPC in controlling non-linear electrical power oscillation damping. With a random tree as the policy, the reinforcement learning is not DRL. With a low-dimensional deterministic model of the system, the authors considered no modeling errors. The results show that with different parameter settings, the reinforcement learning solutions could be worse or better than MPC’s with regard to the cumulative discounted cost. The authors also showed data which indicates that reinforcement learning is at least 10 times faster than MPC during testing.

In [16], the authors compared DRL and receding-horizon control (same as MPC) in controlling a team of unmanned aerial vehicles to maximize wild fire coverage. The authors used a stochastic model of wild fire propagation which adds randomness (disturbances) to the control. The DRL environment state is high-dimensional since it includes both images and continuous states, indicating a hybrid-input DRL control. The results show that DRL outperformed receding-horizon control by a moderate margin regarding cumulative reward.

In [17], the authors compared an integrated MPC-DRL and a pure MPC controllers for control of high-dimensional tasks such as humanoid getup and in-hand manipulation. The integrated MPC-DRL controller is essentially a MPC controller wherein the MPC terminal cost is learned via DRL. The training and testing were based on an accurate model without considering modeling errors. The authors found that the integrated MPC-DRL controller achieved higher rewards than a pure MPC controller by a moderate margin.

In [18], the authors compared DRL and MPC for merging into dense traffic. The DRL and MPC methods do not share the same cost function. Specifically, DRL has a complex cost function including absolute-value costs while MPC has a quadratic cost function. Thus, the authors did not compare the episode costs of DRL and MPC. However, the authors found that the DRL-trained policy significantly out-performed MPC regarding merging success rates.

III. ACC PROBLEM FORMULATION

![Fig. 1. Schematic for two-car following.](image)

In this section, we formulate the ACC problem with state-space equations and define the multi-objective cost function.
We consider controlling a following vehicle \( i \) to maintain a constant time gap \( t_s = 1 \text{s} \) with its immediately preceding vehicle \( i-1 \), see Fig. [1]. The gap-keeping error \( e \) and the velocity difference \( e_v \) between the preceding and following vehicles are:

\[
e = l_{i-1} - l_i - b_{i-1} - (d_0 + hv_i) \\
e_v = v_{i-1} - v_i 
\]

where \( l_{i-1} \) and \( l_i \) are the distances traversed by the preceding \( i-1 \) and following \( i \) vehicles, respectively, \( b_{i-1} \) is the vehicle body length of the preceding vehicle \( i-1 \), \( d_0 \) is a standstill distance for safety, and \( v_{i-1} \) and \( v_i \) are the velocities of the preceding \( i-1 \) and following \( i \) vehicles, respectively. Taking the derivatives of \( e \) and \( e_v \) and using a first-order system to approximate the acceleration dynamics, we obtain the following state-space equations for ACC:

\[
\dot{e} = v_{i-1} - v_i - l_g a_i = e_v - l_g a_i \\
\dot{e}_v = a_{i-1} - a_i = -a_i \\
\dot{a}_i = \frac{u_i - a_i}{\tau}
\]

where \( a_i \) is the acceleration of the following vehicle, \( a_{i-1} \) is the acceleration of the preceding vehicle whose value is set to zero, and \( u_i \) is the control input (commanded acceleration) to the following vehicle. Setting \( a_{i-1} = 0 \) in the state-space equations depicts the situation that the preceding vehicle is running with a constant speed. When the preceding vehicle has varying speeds, its acceleration values can be considered as disturbances. They contribute to the modeling errors and are helpful for us to compare DRL and MPC on handling modeling errors.

The third equation in the state-space equations is the first-order system describing the acceleration dynamics which is also called the longitudinal COM of the following vehicle. The first-order system is a popular choice to describe the behaviors of a Toyota Prius; they also found the control delay to be 0.2s which is what we used in the HFM.

The ACC cost function has three objectives: minimizing gap-keeping error \( e \), control effort \( u_i \), and jerk which is the rate of acceleration \( j_i = \dot{a}_i \). The cost function is defined as:

\[
c = \sum \alpha \sqrt{(\frac{e_i}{e_{\text{max}}})^2 + \varepsilon} + \beta \sqrt{(\frac{u_i}{u_{\text{min}}})^2 + \varepsilon} + \gamma \sqrt{(\frac{\dot{a}_i}{(u_{\text{max}} - u_{\text{min}})/\Delta t})^2 + \varepsilon}
\]

where \( \alpha = 1/3, \beta = 1/3, \) and \( \gamma = 1/3 \) are the weights, \( \varepsilon = 10^{-8} \) is an extremely small value, \( \Delta t = 0.1 \text{s} \) is the time step, \( e_{\text{max}} = 15 \text{m} \) is the nominal maximum gap-keeping error, and \( u_{\text{max}} = 2 \text{m/s}^2 \) and \( u_{\text{min}} = -3 \text{m/s}^2 \) are the allowed maximum and minimum control input values. Table [I] summarizes the ACC parameter values.

**TABLE I**

| Parameter                           | Value |
|-------------------------------------|-------|
| Constant time gap \( t_s \)        | 1s    |
| First-order-system COM time constant \( \tau \) | 0.1s  |
| Control frequency                   | 10Hz  |
| Allowed maximum control input \( u_{\text{max}} \) | 2m/s² |
| Allowed minimum control input \( u_{\text{min}} \) | -3m/s² |
| Nominal maximum gap-keeping error \( e_{\text{max}} \) | 15m   |
| Weight for gap-keeping error \( \alpha \) | 1/3   |
| Weight for control action \( \beta \) | 1/3   |
| Weight for jerk \( \gamma \) | 1/3   |

The cost function is almost the same as an absolute-value cost function except for having \( \varepsilon \). The reason to add the \( \varepsilon \) is to create a smooth and differentiable cost function such that IPO could converge. A pure absolute-value cost function is not differentiable at the minimum which causes IPO unable to converge. A quadratic cost function which is popular for MPC is not used since it causes significant steady-state errors of DRL solutions [24]. We do not use different cost functions for DRL and MPC as it would be meaningless to compare episode costs.

DRL usually utilizes the notion of reward \( r \) which is the negative value of MPC cost \( c \). That is, \( r = -c \). At each time step \( t = 0, 1, 2, ..., T \) where \( T \) is the episode termination time step, the DRL reward \( r_t \) is discounted as \( \gamma^t r_t \) with the discount factor \( \gamma = 0.99 \). Recall that the discount factor is not used in MPC.

**IV. Methodologies**

In this section, the DRL and MPC methodologies to solve the optimal control solutions for ACC are introduced. Since both DRL and MPC are based on discrete time, RungeKutta-4 (RK4) is used to discretize the ACC state-space equations. For both DRL training and MPC optimization, the discrete time step is 0.1s.

**A. DRL**

Reinforcement Learning is formulated as a Markov Decision Process: At time \( t \), given the environment state \( s_t \), an agent takes an action \( a_i \) based on a policy \( \mu \), resulting in a new state \( s_{t+1} \) and a reward \( r_t \). Reinforcement Learning learns
a state-action mapping policy that maximizes the cumulative discounted reward \( \sum_{t=0}^{\infty} \gamma^t r_t \). The specific reinforcement learning algorithm we use, DDPG, has two networks: the actor \( \phi \) and critic \( \theta \) networks. The critic network, also called the Q-network, is trained based on Bellman’s principle of optimality. Specifically, the Q-value for a state-action pair is defined as the cumulative discounted reward from time \( t \): \( Q_\theta(s_t, a_t) = \sum_{t=0}^{\infty} \gamma^t r_t \). The critic network parameters \( \theta \) are updated by minimizing the loss \( L_t = r_t + \gamma Q_\theta(s_{t+1}, \mu_\phi(s_{t+1})) - Q_\theta(s_t, a_t) \) using gradient descent with respect to \( \theta \). The actor network, also called the policy network, is updated by taking a gradient ascent on the Q-value \( Q_\theta(s_t, \mu_\phi(s_t)) \) with respect to \( \phi \).

Several techniques are used to improve training stability and convergence. They include target networks, mini-batch gradient descent, experience replay, batch normalization, and addition of Gaussian noise to the action for exploration \([3, 25]\). Table II shows the DDPG parameter values. Both the actor and critic neural nets have two hidden layers with 64 linear rectifier neurons for each layer. Additionally, before discounting, the reward is clipped to have value within \([-1, 0]\) since it is suggested that sudden large change of reward values decrease training stability \([26]\).

| Parameter                      | Value         |
|--------------------------------|---------------|
| Target network update coefficient | 0.001         |
| Reward discount factor          | 0.99          |
| Actor learning rate             | 0.0001        |
| Critic learning rate            | 0.001         |
| Experience replay memory size   | 500000        |
| Mini-batch size                 | 64            |
| Actor Gaussian noise mean       | 0             |
| Actor Gaussian noise standard deviation | 0.02 |

For training the ACC optimal control policy, the reinforcement learning state evolution is based on the ACC state-space equations with \( a_{i-1} = 0 \). For each training episode, the initial conditions for the state \( e_0, e_r, 0, a_{i,0} \) are randomly distributed in \([-5, 5], [-5, 5], \) and \([-3, 2]\), respectively. All the training episode length is 20s. We trained the DDPG policy for 1.2 million time steps and observed convergence of the cumulative reward. The undiscounted cumulative reward during the training is shown in Fig. 2. The reason to show the undiscounted cumulative reward is that changes of the undiscounted cumulative reward are more observable during training. The training took about an hour on a desktop computer with a 16-core (32-thread) AMD processor and a Nvidia GeForce RTX GPU.

### B. MPC

With the ACC state-space equations and the cost function, we formulate the optimization problem using direct single shooting for a certain prediction horizon. Then the optimization problem is solved using IPO. The first value of the solved control sequence is applied. Such procedure is repeated as the receding-horizon MPC. Note that we also solve for the optimization solution just once for the entire episode using IPO as the benchmark solution. The benchmark is denoted as IPO solution for the rest of the paper. The MPC optimization problem is formulated and solved using the open-source symbolic framework CasADi \([27]\).

### V. RESULTS

This section shows the testing results of the DRL and MPC controllers without and with modeling errors. For without modeling errors, the controllers are tested on the COM with a fixed initial condition and \( a_{i-1} = 0 \). COM testing allows us to better understand the inherent differences of DRL and MPC before the effect of modeling errors. For with modeling errors, the controllers are tested on the HFM with the same fixed initial condition and \( a_{i-1} = 0 \) as for the COM. This allows us to understand how modeling errors may degrade the DRL and MPC performances differently. We also conducted HFM testing with the drive cycles which is more realistic. In this case, the preceding vehicle’s velocity profile follows the drive cycles, meaning that \( a_{i-1} \) is not zero. For drive cycle testing, the modeling errors are larger due to non-zero \( a_{i-1} \).

#### A. COM testing with constant speed following

Without modeling errors, the DRL and MPC controllers are tested on the COM. The fixed initial condition is \([e_0, e_{r,0}, a_{i,0}]=[5m, 5m/s, 0m/s]^2\]. With \( a_{i-1} = 0 \), the preceding vehicle has a constant velocity. The results of the episode is plotted in Fig. 3. The curves for the three methods, DRL, MPC, and IPO, look similar. In the third plot of control input and acceleration values, the DRL curve exhibits an uncommon shape, indicating that the trained policy is not formula-based. The plotted MPC solution has a prediction horizon \( h=2.8s \), which is adequate but not comparably long. For \( h=2.8s \), the MPC solution decelerates faster and with larger values compared to the benchmark IPO solution.

The MPC episode cost decreases with increasing prediction horizon, see Fig. 4 and Table III. There is an acute change of the episode cost when increasing the prediction horizon from 2.7s to 2.8s, see Table III. This may be because the defined cost function which is almost the same as an absolute-value cost function, which generates acute MPC performance improvement around a certain prediction horizon. We experimented with a quadratic cost function whose results are not
Fig. 3. (Color Online) COM test results of an episode. Note that when MPC $h=5$s, the MPC results are almost identical to those of IPO.

Fig. 4. (Color Online) Episode cost simulation time in python for COM testing.

|               | Episode cost increase compared to the IPO solution [%] | Episode simulation time in python [s] |
|---------------|------------------------------------------------------|--------------------------------------|
| DRL           | 4.19                                                 | 0.4518                               |
| MPC ($h=2.5s$)| 1413.08                                              | 99.01                                |
| MPC ($h=2.7s$)| 1370.34                                              | 70.10                                |
| MPC ($h=2.8s$)| 2.16                                                 | 65.82                                |
| MPC ($h=3s$)  | 1.42                                                 | 76.05                                |
| MPC ($h=5s$)  | -0.02                                                | 181.68                               |
| IPO           | 0                                                    | 50.90                                |

TABLE III

Comparison of DRL, MPC, and IPO solutions when testing on the COM.

shown here, the transition of the episode cost with increasing prediction horizon values are smoother and slower.

From Table III, the DRL episode cost is 4.19% higher than that of the benchmark IPO solution, while MPC with a long prediction horizon $h=5$s is equivalent to IPO (just 0.02% lower). This suggests that DRL is equivalent to MPC with a sufficiently long prediction horizon with regards to the episode cost. From Table III we also see that the DRL-trained policy which is a neural net took less than 0.5 seconds in episode testing while the MPC and IPO optimization methods took at least 50 seconds, given that all testing was carried out in python on the same computer.

B. HFM testing with constant speed following

With modeling errors, DRL and MPC controllers are tested on the HFM. Remember that the first-order-system COM is an approximation to the acceleration dynamics of the HFM, which generates modeling errors. The modeling errors for the 2015 Toyota Prius hybrid electric vehicle that we consider have multiple causes. Firstly, the control delay of 0.2s is not considered in the COM. Secondly, the COM does not model mode switches between electric and hybrid modes of Prius which can happen during speeding and braking. Thirdly, the low-level PI controller causes overshoot and oscillation when correcting the error between the control input and actual acceleration, which is also not modeled in the COM. Last but not least, the COM does not model the power limit of the vehicle which could happen at high speeds. The power limit contributes to the largest modeling errors as it happens. There, we consider HFM testing at different (low and high) speeds.

The state initial condition for the HFM testing is the same as for the COM testing. The preceding vehicle also has a constant
speed with $a_{t-1} = 0$. During simulation, RK4 is used for HFM discretization. Note that the HFM operates at 100Hz contrary to training or control frequency 10Hz (0.1s as the time step) of the COM.

**TABLE IV**

| $v_{i,0}$ [m/s] | MPC ($h=5s$) episode cost increase compared to the DRL solution [%] |
|----------------|---------------------------------------------------------------|
| 0              | 1.30                                                          |
| 5              | 1.21                                                          |
| 10             | -0.55                                                         |
| 15             | 0.73                                                          |
| 20             | 19.92                                                         |
| 25             | 13.94                                                         |

The first two rows of plots in Fig. 5 show the HFM testing results when the initial speeds of the following vehicle $v_{i,0}$ are 5m/s and 25m/s, respectively. Correspondingly, the preceding vehicle is running at 10m/s and 30m/s, respectively. For $v_{i,0}=5$m/s, the DRL and MPC results look similar. In fact, MPC episode cost is just 1.3% higher than that of DRL, see Table [IV]. For $v_{i,0}=25$m/s, the following vehicle experienced the power limit constraint, see the third plot of
In this work, we compare DRL and MPC performances for ACC car-following control. The DRL training and MPC optimization are based on the same state-space equations and cost function. They also share the same discretization method and training and control frequency. We also keep the testing conditions the same for DRL and MPC. The only difference is that DRL utilizes a reward discount factor $\gamma = 0.99$ for convergence purpose. The discount factor is not used for MPC since it is abnormal to use it. Our results show that, without modeling errors (testing on the COM), DRL is equivalent to MPC with a sufficiently long prediction horizon. Additionally, the DRL episode cost is just 4.19% higher than that of the benchmark IPO solution. It suggests that DRL can be used to train a sub-optimal control policy that is very close to the optimal. With modeling errors (testing on the HFM and with drive cycles), DRL is significantly better than MPC regarding episode costs when the modeling errors are significant. When the modeling errors are small, the DRL and MPC performances are similar. DRL is more tolerant with smaller cost increase than MPC in the presence of the same modeling errors.

Despite the advantages of DRL, one could always use an accurate model to reduce the modeling errors and significant degradation of the MPC performance. Recent developments in tube-based MPC could also make MPC more robust and disturbance-tolerant [12]. For DRL, the use of transfer learning or meta learning on the trained policy could make it even better in handling modeling errors and uncertainties [23], [29]. Also, the ACC car-following control is a low-dimensional task with only three state variables. DRL is known to handle well higher-dimensional tasks with complex cost functions [6], [17]. Thus, it would be interesting to compare DRL and MPC with these factors considered.

One of the known short comings of MPC is the computation time it needs for online optimization. One of the known disadvantages of DRL is the lack of theoretical guarantees for the black-box neural-net solution. The combination of both could potentially alleviate the short comings of both and make use of their advantages. In fact, such research efforts are ongoing [17], [30], [31], [32], [33].

**ACKNOWLEDGMENT**

The authors would like to thank Toyota, Ontario Centres of Excellence, and the Natural Sciences and Engineering Research Council of Canada for the support of this work.

**REFERENCES**

[1] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.
[2] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot et al., “Mastering the game of go with deep neural networks and tree search,” *Nature*, vol. 529, no. 7587, p. 484, 2016.
[3] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski et al., “Human-level control through deep reinforcement learning,” *Nature*, vol. 518, no. 7540, p. 529, 2015.
[4] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, “Trust region policy optimization,” in *International conference on machine learning*, 2015, pp. 1889–1897.
[5] J. Schulman, F.沃尔斯基, P. Dhariwal, A. Radford, and O. Klimov, “Proximal policy optimization algorithms,” *arXiv preprint arXiv:1707.06347*, 2017.
Fig. 7. (Color Online) HWFET drive cycle testing results.

### TABLE VI

| Drive cycle | Method | $e_{\text{min}}$, $e_{\text{mean}}$, $e_{\text{max}}$ | $J_{\text{min}}$, $J_{\text{mean}}$, $J_{\text{max}}$ | MPC episode cost increase compared to DRL results [%] |
|-------------|--------|---------------------------------|---------------------------------|---------------------------------|
| HWFET       | DRL    | -0.25, -0.04, 0.27              | -2.55, 0, 3.09                  | -                               |
|             | MPC    | -0.64, 0, 0.64                  | -13.65, 0, 9.91                 | 49.62                           |
| FTP-75      | DRL    | -0.26, -0.02, 0.54              | -3.29, 0, 4.37                  | -                               |
|             | MPC    | -0.81, 0.01, 1.45               | -25.96, 0, 20.81                | 30.2                            |
| US06        | DRL    | -2.09, 0.60, 21.18              | -14.65, 0, 13.71                | -                               |
|             | MPC    | -1.18, 0.67, 21.80              | -31.52, 0, 19.33                | 41.22                           |

[6] T. P. Lillicrap, J. J. Hunt, A. Prizel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, “Continuous control with deep reinforcement learning,” arXiv preprint arXiv:1506.02971, 2015.

[7] J. A. Rossiter, Model-based predictive control: a practical approach. CRC press, 2017.

[8] M. Diehl, H. J. Ferreau, and N. Haverbeke, “Efficient numerical methods for nonlinear mpc and moving horizon estimation,” in Nonlinear model predictive control. Springer, 2009, pp. 391–417.

[9] M. Vajedi and N. L. Azad, “Ecological adaptive cruise controller for plug-in hybrid electric vehicles using nonlinear model predictive control,” IEEE Transactions on Intelligent Transportation Systems, vol. 17, no. 1, pp. 113–122, 2015.

[10] Y. Lin, J. McPhee, and N. L. Azad, “Longitudinal dynamic versus kinematic models for car-following control using deep reinforcement learning,” arXiv preprint arXiv:1905.08314, 2019.

[11] B. Sakhdari and N. L. Azad, “Adaptive tube-based nonlinear mpc for economic autonomous cruise control of plug-in hybrid electric vehicles,” IEEE Transactions on Vehicular Technology, vol. 67, no. 12, pp. 11 390–11 401, 2018.

[12] B. T. Lopez, J. P. Howl, and J.-J. E. Slotine, “Dynamic tube mpc for nonlinear systems,” in 2019 American Control Conference (ACC). IEEE, 2019, pp. 1655–1662.

[13] A. Eskandarian, Handbook of Intelligent Vehicles. Springer London, 2012.

[14] A. Taghavipour, R. Masoudi, N. L. Azad, and J. McPhee, “High-fidelity modeling of a power-split plug-in hybrid electric powertrain for control performance evaluation,” in ASME 2013 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference. American Society of Mechanical Engineers Digital Collection, 2014.

[15] D. Ernst, M. Glavic, F. Capitanescu, and L. Wenhenkel, “Reinforcement learning versus model predictive control: a comparison on a power system problem,” IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), vol. 39, no. 2, pp. 517–529, 2008.

[16] K. D. Julian and M. J. Kochenderfer, “Distributed wildfire surveillance with autonomous aircraft using deep reinforcement learning,” Journal of Guidance, Control, and Dynamics, pp. 1–11, 2019.

[17] K. Lowrey, A. Rajeswaran, S. Kakade, E. Todorov, and I. Mordatch, “Plan online, learn offline: Efficient learning and exploration via model-based control,” arXiv preprint arXiv:1811.01848, 2018.

[18] D. M. Saxena, S. Bae, A. Nakhaei, K. Fujimura, and M. Likhachev, “Driving in dense traffic with model-free reinforcement learning,” arXiv preprint arXiv:1909.06710, 2019.

[19] S. Li, K. Li, R. Rajamani, and J. Wang, “Model predictive multi-objective vehicular adaptive cruise control,” IEEE Transactions on Control Systems Technology, vol. 19, no. 3, pp. 556–566, 2010.

[20] J. Ploeg, B. T. Scheepers, E. Van Nunen, N. Van de Wouw, and H. Nijmeijer, “Design and experimental evaluation of cooperative adaptive cruise control,” in 2011 14th International IEEE Conference on Intelligent Transportation Systems (ITSC). IEEE, 2011, pp. 260–265.

[21] R. Kianfar, B. Augusto, A. Ebadighajari, U. Hakeem, J. Nilsson, A. Raza, R. S. Tabar, N. V. Irukulapati, C. England, P. Falcone et al., “Design and experimental validation of a cooperative driving system in the grand cooperative driving challenge,” IEEE transactions on Intelligent Transportation Systems, vol. 13, no. 3, pp. 994–1007, 2012.

[22] L. Guvene, I. M. C. Uyan, K. Kahraman, R. Karahmetoglu, I. Altay, M. Senturk, M. T. Emirler, A. E. H. Karci, B. A. Guvene, E. Altug et al., “Cooperative adaptive cruise control implementation of team mekar at the grand cooperative driving challenge,” IEEE Transactions on Intelligent Transportation Systems, vol. 13, no. 3, pp. 1062–1074, 2012.

[23] S. E. Li, Y. Zheng, K. Li, Y. Wu, J. K. Hedrick, F. Gao, and H. Zhang, “Dynamical modeling and distributed control of connected and automated vehicles: Challenges and opportunities,” IEEE Intelligent Transportation Systems Magazine, vol. 9, no. 3, pp. 46–58, 2017.

[24] J.-M. Engel and R. Babuška, “On-line reinforcement learning for non-linear motion control: Quadratic and non-quadratic reward functions,” IFAC Proceedings Volumes, vol. 47, no. 3, pp. 7043–7048, 2014.

[25] S. Ioffe and C. Szegedy, “Batch normalization: Accelerating deep network training by reducing internal covariate shift,” arXiv preprint arXiv:1502.03167, 2015.

[26] H. P. van Hasselt, A. Guez, M. Hessel, V. Mnih, and D. Silver, “Learning values across many orders of magnitude,” in Advances in Neural Information Processing Systems, 2016, pp. 4287–4295.

[27] J. A. E. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl, “CasADi – A software framework for nonlinear optimization and optimal control,” Mathematical Programming Computation, In Press, 2018.

[28] J. van Baar, A. Sullivan, R. Cordorel, D. Jha, D. Romeres, and D. Nikovski, “Sim-to-real transfer learning using robustified controllers for fast adaptation of deep networks,” in Proceedings of the 34th International Conference on Machine Learning-Volume 70. JMLR. org, 2017, pp. 1126–1135.

[29] G. Williams, N. Wagener, B. Goldfain, P. Drews, J. M. Rehg, B. Boots, and E. A. Theodorou, “Information theoretic mpc for model-based reinforcement learning,” in 2017 IEEE International Conference on Robotics and Automation (ICRA). IEEE, 2017, pp. 1714–1721.

[30] S. Kamthe and M. P. Deisenroth, “Data-efficient reinforcement learning versus model predictive control: a comparison on a power system problem,” IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), vol. 39, no. 2, pp. 517–529, 2008.
learning with probabilistic model predictive control,” *arXiv preprint arXiv:1706.06491*, 2017.

[32] A. Nagabandi, G. Kahn, R. S. Fearing, and S. Levine, “Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning,” in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 7559–7566.

[33] S. Gros and M. Zanon, “Data-driven economic mpc using reinforcement learning.” *IEEE Transactions on Automatic Control*, 2019.

**Yuan Lin** received the B.E. degree in Civil Engineering from Nanchang University, China, in 2011 and the Ph.D. degree in Engineering Mechanics from Virginia Tech, Blacksburg, VA, USA, in 2016. He was a Postdoctoral Fellow in the Mechanical Engineering Department at Virginia Tech from 2016 to 2018. He is currently a Postdoctoral Fellow in the Systems Design Engineering Department at the University of Waterloo. His research interests include control, reinforcement learning, autonomous driving, and multi-agent systems.

**John McPhee** is a Professor and the Canada Research Chair in System Dynamics at the University of Waterloo, Canada, which he joined in 1992. Prior to that, he held fellowships at Queens University, Canada, and the Université de Liège, Belgium. He pioneered the use of linear graph theory and symbolic computing to create real-time models and model-based controllers for multi-domain dynamic systems, with applications ranging from autonomous vehicles to rehabilitation robots and sports engineering. His research algorithms are a core component of the widely-used MapleSim modelling software, and his work appears in over 130 journal publications. Prof. McPhee is the past Chair of the International Association for Multibody System Dynamics, a co-founder of 2 international journals and 3 technical committees, a member of the Golf Digest Technical Panel, and an Associate Editor for 5 journals. He is a Fellow of the Canadian Academy of Engineering, the American and Canadian Societies of Mechanical Engineers, and the Engineering Institute of Canada. He has won 8 Best Paper Awards and, in 2014, he received the prestigious NSERC Synergy Award from the Governor-General of Canada.

**Nasser L. Azad** is currently an Associate Professor in the Department of Systems Design Engineering, University of Waterloo, and the Director of Smart Hybrid and Electric Vehicle Systems (SHEVS) Laboratory. He was a Postdoctoral Fellow with the Vehicle Dynamics and Control Laboratory, Department of Mechanical Engineering, University of California, Berkeley, CA, USA. Dr. Azad’s primary research interests lie in control of connected hybrid and electric vehicles, autonomous cars, and unmanned aerial vehicle quad-rotors. He is also interested in applications of Artificial Intelligence for solving different engineering problems. Due to his outstanding work, Dr. Azad received an Early Researcher Award in 2015 from the Ministry of Research and Innovation, Ontario, Canada.