processes

MDPI

Article

Intelligent Control Strategy for Transient Response of a Variable Geometry Turbocharger System Based on Deep Reinforcement Learning

Bo Hu [1,2,*], Jie Yang [1], Jiaxi Li [1], Shuang Li [1] and Haitao Bai [1]

[1] Key Laboratory of Advanced Manufacturing Technology for Automobile Parts, Ministry of Education, Chongqing University of Technology, Chongqing 400054, China; j.young@2017.cqut.edu.cn (J.Y.); 11607990404@2016.cqut.edu.cn (J.L.); ls123456@2016.cqut.edu.cn (S.L.); Baiht@2017.cqut.edu.cn (H.B.)

[2] State Key Laboratory of Engines, Tianjin University, Tianjin 300072, China

* Correspondence: b.hu@cqut.edu.cn

Received: 15 August 2019; Accepted: 2 September 2019; Published: 6 September 2019

Abstract: Deep reinforcement learning (DRL) is an area of machine learning that combines a deep learning approach and reinforcement learning (RL). However, there seem to be few studies that analyze the latest DRL algorithms on real-world powertrain control problems. Meanwhile, the boost control of a variable geometry turbocharger (VGT)-equipped diesel engine is difficult mainly due to its strong coupling with an exhaust gas recirculation (EGR) system and large lag, resulting from time delay and hysteresis between the input and output dynamics of the engine's gas exchange system. In this context, one of the latest model-free DRL algorithms, the deep deterministic policy gradient (DDPG) algorithm, was built in this paper to develop and finally form a strategy to track the target boost pressure under transient driving cycles. Using a fine-tuned proportion integration differentiation (PID) controller as a benchmark, the results show that the control performance based on the proposed DDPG algorithm can achieve a good transient control performance from scratch by autonomously learning the interaction with the environment, without relying on model supervision or complete environment models. In addition, the proposed strategy is able to adapt to the changing environment and hardware aging over time by adaptively tuning the algorithm in a self-learning manner on-line, making it attractive to real plant control problems whose system consistency may not be strictly guaranteed and whose environment may change over time.

Keywords: self-learning; transient response; variable geometry turbocharger; deep reinforcement learning; deep deterministic policy gradient

1. Introduction

The concept of engine downsizing and down-speeding enables reductions in fuel consumption and CO_2 emissions from passenger cars in order to satisfy the greenhouse gas emission reduction targets set by the 2015 Paris Climate Change Conference [1,2]. These reductions are achieved by reducing pumping and friction losses at part-load operation. Conventionally, rated torque and power for downsized units are recovered by means of fixed-geometry turbocharging [3]. The transient response of such engines is, however, affected by the static and dynamic characteristics of the fixed-geometry turbo-machinery (especially when it is optimized for high-end torque) [4,5]. One feasible solution to this is the use of variable geometry turbocharger (VGT) technology, which is designed to enable the effective aspect ratio of the turbocharger to vary with different engine operating conditions (see Figure 1). This is because the best aspect ratio at high engine speeds is very different from the best aspect ratio at low engine speeds [6]. In engines equipped with VGT, and because part of the exhaust

energy is used to accelerate the turbine shaft for boosting, engine transient response and fuel economy can be improved significantly [7].

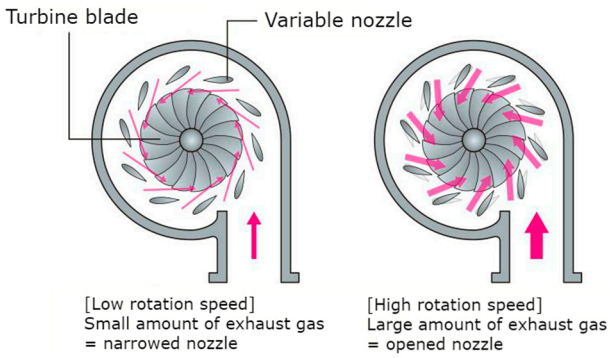

Figure 1. Variable geometry turbocharger (VGT) operating principle.

For diesel engines, VGT often interacts with an exhaust gas recirculation (EGR) system (which is for the engine's NOx emission reduction). This interaction increases the complexity of the VGT control problem. Furthermore, the time delay and hysteresis between the input and output dynamics of the diesel engine's gas exchange system make it difficult to accurately control the VGT [8]. Traditionally, the fixed-parameter structure proportion integration differentiation (PID) control is used in industry for VGT boost control, but the parameter setting processing is complicated and it is difficult to obtain satisfactory results, especially when the state of the control loop is altered [9–13]. There are other PID variants that include expert PID control [14], fuzzy PID control [15], and neural network-based PID control [16], etc. Although the variants are said to perform better if tuned well, those algorithms, respectively, need to acquire expert knowledge, construct fuzzy control decision tables, and tune complicated neural network parameters, and thus their widespread use for VGT boost control may be prohibited. Meanwhile, for complex industrial systems with high-order, large lag, strong coupling, nonlinear, and time-varying parameters (such as for VGT control systems [17–19]), the traditional control theory which relies on mathematical models is still immature, and some methods are too complicated and cannot be directly applied for industrial applications [20–22]. On the other hand, it may not be possible or feasible to develop a first-principle model for complex industrial processes. Furthermore, complex engineering systems are rather expensive, with a high requirement for system reliability and control performance. In this context, "model-free" intelligent algorithms (in the absence of a model with high fidelity) to achieve end-to-end learning and intelligent control while taking the industrial need of simplicity and robustness into consideration may provide an attractive alternative.

Reinforcement learning (RL), which is considered as one of three machine learning paradigms, focuses on how agents should act in the environment to maximize cumulative rewards (see Figure 2) [20]. Temporal-difference (TD) learning, which is a combination of dynamic programming (DP) ideas and the Monte Carlo idea, is considered to be the core and novelty of reinforcement learning [23]. In RL, there are two classes of the TD method—on-policy and off-policy. The most important on-policy algorithm includes Sarsa and Sarsa (λ), and one of the breakthroughs in off-policy reinforcement learning is known as Q-learning [24,25]. Deep reinforcement learning (DRL) is an area of machine learning that combines a deep learning approach and reinforcement learning (RL). This field of study was used to master a wide range of Atari 2600 games and its great success on AlphaGo, which was the first computer program to beat a human professional Go player, is a historic milestone in machine learning research [26]. Deep Q-network (DQN) based on value function and deep deterministic policy gradient (DDPG) based on policy gradient are the two latest DRL techniques. The DQN used on AlphaGo and AlphaGo Zero [27,28] uses only the original image of the game as input, and does not rely on the manual extraction of features. It innovatively combines deep convolutional neural networks with Q-learning to achieve human player control (it also achieved great success in Atari

video games [29]). Although this algorithm achieves the generalization of continuous state space, it is theoretically only suitable for tasks in discrete action space. The DDPG strategy proposed by Lillicrap et al. [30] uses deep neural networks as approximators to effectively combine deep learning and deterministic strategy gradient algorithms [31]. It can cope with high-dimensional inputs, achieve end-to-end control, output continuous actions, and thus can be applied to more complex situations with large state spaces and continuous action spaces. To the authors' best knowledge, there is no literature that has applied DRL techniques to boost control problems for VGT-equipped engines. Furthermore, there seem to be few studies that analyze the DDPG algorithm on sequential decision control problems for industrial applications.

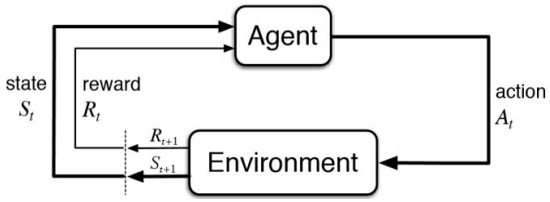

Figure 2. Basic idea and elements involved in a reinforcement learning formalization.

Based on the above discussion, it is appropriate to apply the DDPG techniques for the boost control on a VGT-equipped engine. In this paper, in order to achieve the optimum boost control performance, first the simulation model of a VGT-equipped diesel engine is introduced. Subsequently, a model-free DDPG algorithm is built to develop and finally form a strategy to track the target engine boost pressure under transient driving cycles by regulating the turbine vanes. Finally, the proposed DDPG algorithm is compared with a fine-tuned PID controller to validate its optimality. The rest of this article is structured as follows: Section 2 describes the mean value engine model (MVEM) of the VGT-equipped diesel engine. In Section 3, the DDPG-based framework is proposed to achieve the optimal boost control of the engine. In Section 4, the corresponding simulations are conducted to compare the proposed algorithm and a fine-tuned PID controller. Section 5 concludes the article.

2. Mean Value Engine Model Analysis

Mean value engine models (MVEMs) are useful for certain types of modeling where simulation speed is of primary importance, the details of wave dynamics are not critical, and bulk fluid flow is still important (for modeling turbocharger lag, etc.) [32]. A mean value engine model essentially contains a map-based cylinder model, which is computationally faster than a detailed cylinder. The simulation speed can be increased further by combining multiple detailed cylinders into a single mean value cylinder. In addition, many of the other flow components from the detailed model can be combined to create a simplified flow network of larger volumes.

The layout of the VGT-equipped diesel engine is illustrated in Figure 3. The detailed engine model was converted to a mean value model. As it is not the focus of this article, only a brief process summary is presented here. The mean value cylinder is defined by imposed values for indicated mean effective pressure (IMEP), volumetric efficiency, and exhaust gas temperature. These three quantities are predicted by neural networks (see Figure 4) that depend on seven input variables (intake manifold pressure and temperature, exhaust manifold pressure, EGR fraction, injection timing, fuel rate, and engine speed). Note here that each neural network (four in total) is trained using the data generated in the detailed simulation. The output of this training is an external file which contains the necessary neural network settings. Once the training has been completed, the neural network file can be called into the mean value model, which dramatically increases the computational efficiency. In addition, the friction mean effective pressure (FMEP) of the cranktrain is also calculated by a neural network dependent on the same seven variables.

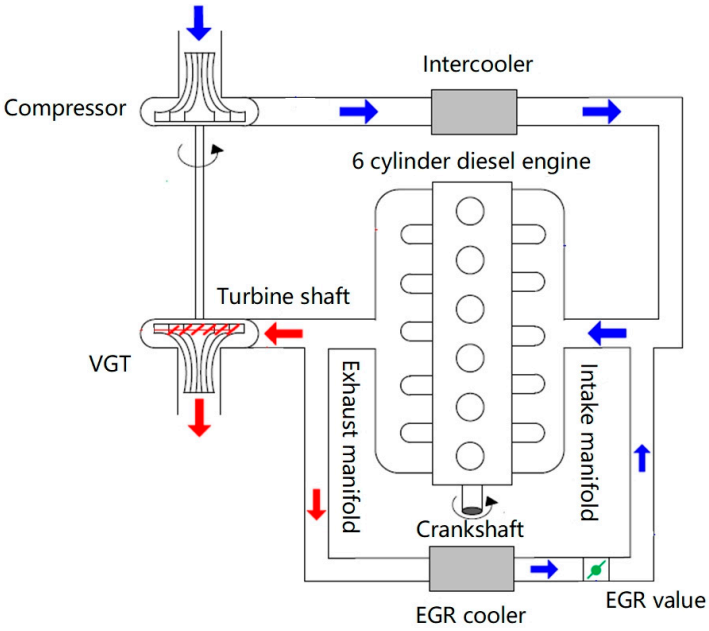

Figure 3. VGT-equipped diesel engine with an exhaust gas recirculation (EGR) system.

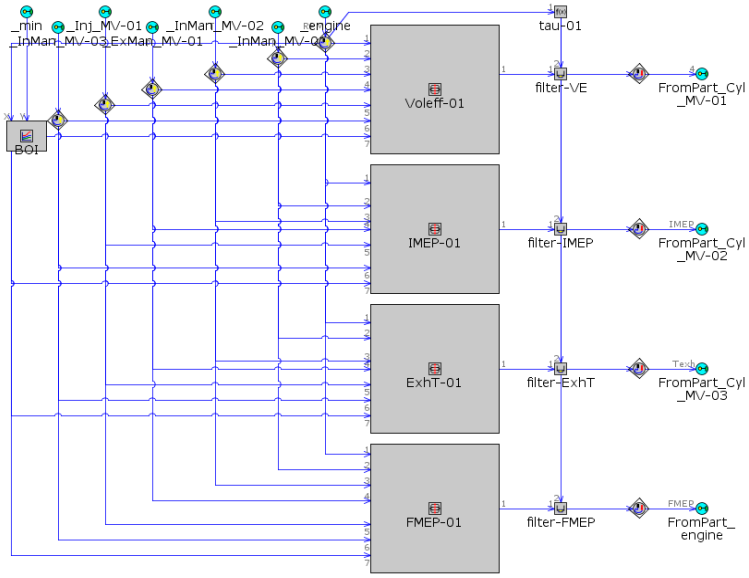

Figure 4. The proposed neural network that maps the mean value cylinder performance as a function of seven input variables in GT-SUITE.

The intake and exhaust systems are simplified into large "lumped" volumes so that system volume is conserved (with a loss of detailed wave dynamics). The large volumes allow large time steps to be taken by the solver. Pressure drops in the flow network are calibrated using restrictive orifice connections between the lumped volumes. Additionally, heat transfer rates are calibrated using the heat transfer multiplier in parts where heat transfer is significant (exhaust manifold). The intercooler and EGR cooler outlet temperatures are imposed, which allows the gas temperature to be imposed as it passes through the connection. This reduces the amount of volumes required and allows a reduction in potential for any instability in the solver caused by the high heat transfer rates in the heat exchangers at large time steps that are typical of mean value models. The mean value model results match the detailed results well (see Figure 5), and should provide sufficient accuracy for control system and vehicle transient studies. The mean value model runs approximately 150 times faster than the detailed

model and runs faster than "real time", enabling it to be used for real-time hardware-in-the-loop (HIL) simulation.

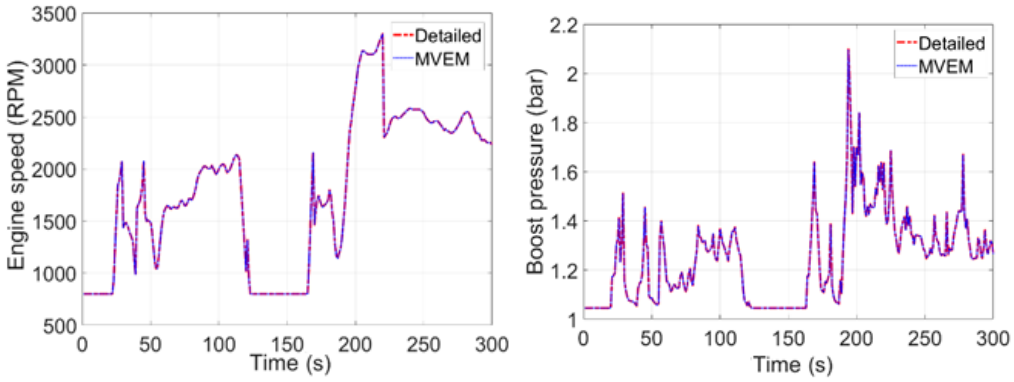

Figure 5. The first 300 s engine speed and boost pressure comparison for the US EPA FTP-72 (Federal Test Procedure) drive cycle.

The research engine was a 6 cylinder 3 L turbocharged direct injection (DI) diesel, with its GT-SUITE model seen in Figure 6. Advanced controllers should be used to dynamically control the position of the VGT rack in order to achieve the target boost pressure. It should be noted that the model was initially controlled by a fine-tuned PID controller, and the target boost pressure and the P and I gains were both mapped as a function of speed and requested load (implied by accelerator pedal position).

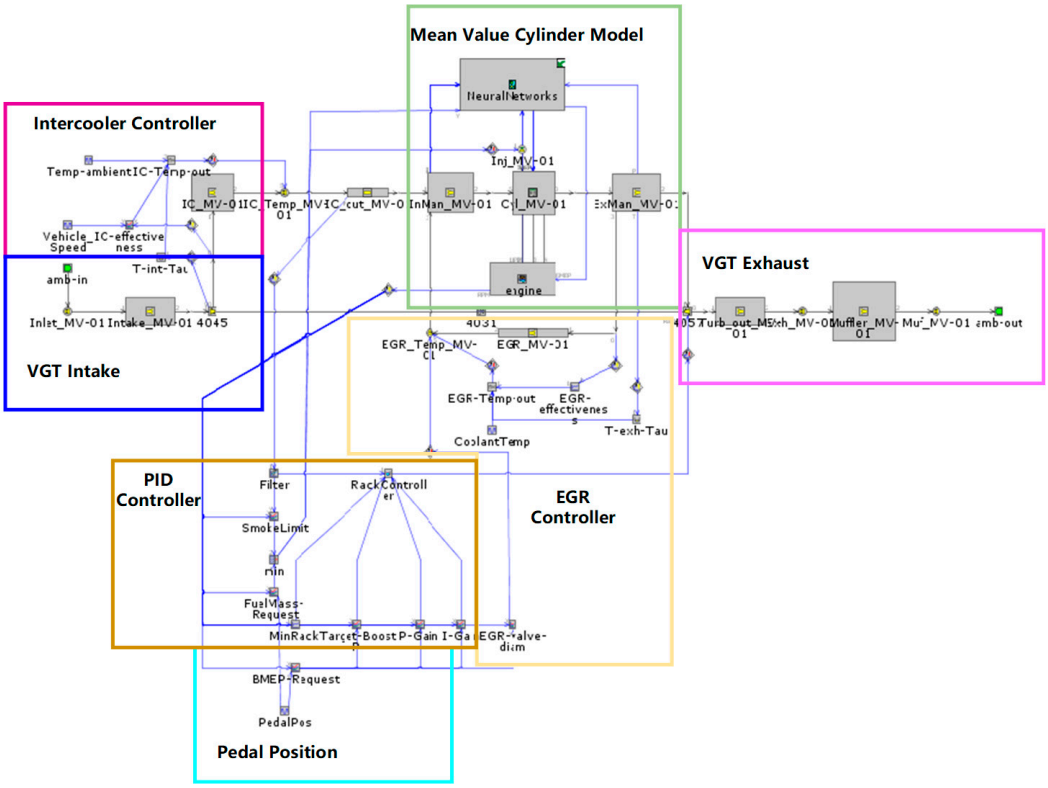

Figure 6. The GT-SUITE model layout of the 6 cylinder 3 L VGT-equipped diesel.

To analyze the transient behavior, the engine speed was imposed to match the prescribed vehicle speed profile from the FTP-72 driving cycle (see Figure 7). This transient engine speed profile was

calculated using a simple kinematic mode simulation which can be seen in Figure 8. The same simulation provided the required brake mean effective pressure (BMEP) from the engine. Then, a separate detailed simulation was run with an injection controller to determine and store the transient pedal position required to achieve the requested BMEP.

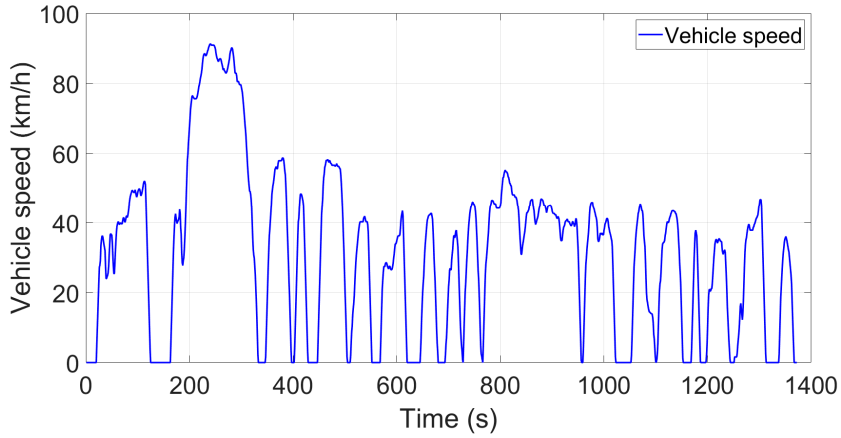

Figure 7. FTP-72 driving cycle.

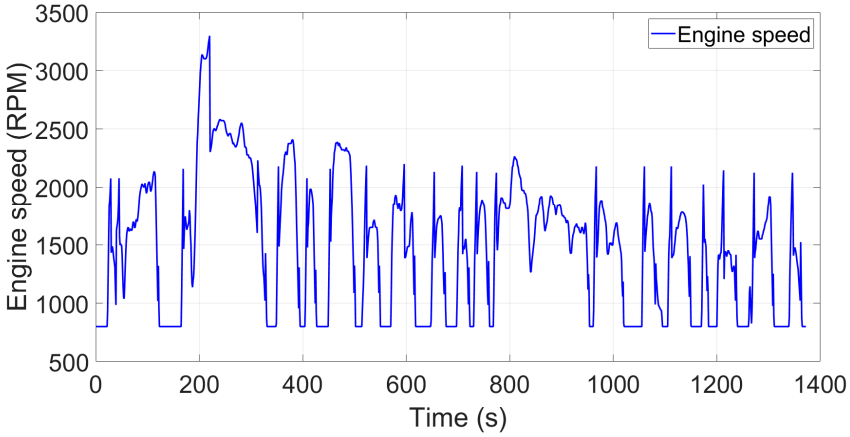

Figure 8. FTP-72 transient engine speed profile.

3. Deep Reinforcement Learning Algorithm

Model-free reinforcement learning is a technique for understanding and automating goal-directed learning and decision-making [33]. It differs from most other control algorithms in that it emphasizes on agents learning through direct interaction with the environment, without relying on model supervision or a complete environmental model [34]. As an interactive learning method, the main features of reinforcement learning are trial-and-error search and delayed return [35]. Figure 1 shows the interaction process between the agent and the environment. At any time step, the agent observes the environment in order to get the state S_t and then performs the action A_t. Afterward, the environment generates the next time S_{t+1} and R_t according to A_t. The probability that the process moves into its new state S_{t+1} is influenced by the chosen action and is given by the state transition function. Such a process can be described by Markov decision processes (MDPs) [36,37]. The goal of reinforcement learning is to formulate the problem as an MDP and find the optimal strategy [38]. The so-called strategy refers to the state-to-action mapping, which commonly uses the symbol policy π. It refers to a mapping on

the action set for a given state s, that is, $\pi(a|s) = p[A_t = a|S_t = s]$. Reinforcement learning introduces a reward function to represent the return value at a certain time t, as follows:

$$G_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}, \quad (1)$$

where r represents an immediate reward and γ represents a discount factor which shows how important future returns are relative to current returns. The action–value function used in reinforcement learning algorithms describes the expected return after taking an action in state S_t and thereafter following policy π:

$$Q^{\pi}(s_t, a_t) = \sum_{r_{i\geq t}, s_{i>t} \sim E, a_{i>t} \sim \pi} [R_t | s_t, a_t] \quad (2)$$

Reinforcement learning makes use of the recursive relationship known as the Bellman equation:

$$Q^{\pi}(s_t, a_t) = \sum_{r_t, s_{t+1} \sim E} [r(s_t, a_t) + \gamma \sum_{a_{t+1} \sim \pi} [Q^{\pi}(s_{t+1}, a_{t+1})]] \quad (3)$$

The expectation depends only on the environment, which means that it is possible to learn Q^{μ} off-policy using transitions that are generated from a different stochastic behavior policy β. Q-learning, a commonly used off-policy algorithm, uses the greedy policy $\mu(s) = \text{argmax}_a\, Q(s,a)$. We consider function approximators parameterized by θ^Q, which we optimize by minimizing the loss, as follows:

$$L\left(\theta^Q\right) = \sum_{s_t \sim \rho^{\beta}, a_t \sim \beta, r_t \sim E} [\left(Q\left(s_t, a_t | \theta^Q\right) - y_t\right)^2] \quad (4)$$

where

$$y_t = r(s_t, a_t) + \gamma Q(s_{t+1}, \mu(s_{t+1}) | \theta^Q)$$

Recently, deep Q-network (DQN) adapted the Q-learning algorithm in order to make effective use of large neural networks as function approximators. Before DQN, it was generally deemed that it was difficult and unstable to use large nonlinear function approximators for learning value functions. Thanks to two innovations, DQN can use a function approximator to learn the value function in a stable and robust manner: (a) the network is trained off-policy with samples from a replay buffer to minimize correlations between samples and (b) the target Q-network is used to train the network to provide consistent goals during time difference (TD) backups.

The deterministic policy gradient (DPG) algorithm maintains a parameterized actor function, $\mu(s|\theta^{\mu})$, which specifies the current policy by deterministically mapping states to a specific action. The critic $Q(s,a)$ is learned using the Bellman equation as in Q-learning. The actor is updated by applying the chain rule to the expected return from the start distribution J with respect to the actor parameters, as follows:

$$\begin{aligned} \nabla_{\theta^{\mu}} J &\approx \sum_{s_t \sim \rho^{\beta}} [\nabla_{\theta^{\mu}} Q(s, a | \theta^Q)|_{s=s_t, a=\mu(s_t|\theta^{\mu})}] \\ &= \sum_{s_t \sim \rho^{\beta}} [\nabla_a Q(s, a | \theta^Q)|_{s=s_t, a=\mu(s_t)} \nabla_{\theta_{\mu}} \mu(s | \theta^{\mu})|_{s=s_t}] \end{aligned} \quad (5)$$

Deep deterministic policy gradient (DDPG) combines the actor–critic (AC) approach based on deterministic policy gradient (DPG) [31] with insights from the recent success of deep Q-network (DQN). Although DQN has achieved great success in high-dimensional issues, like the Atari game, the action space for which the algorithm is implemented is still discrete. However, for many tasks of interest, especially physical industrial control tasks, the action space must be continuous. Note that if the action space is discretized too finely, the control problem eventually leads to excessive motion space, which is extremely difficult for the algorithm to learn. The DDPG strategy uses deep neural networks as approximators to effectively combine deep learning and deterministic strategy gradient algorithms. It can cope with high-dimensional inputs, achieve end-to-end control, output continuous actions, and thus can be applied to more complex situations with large state spaces and continuous action spaces. In detail, DDPG uses an actor network to tune the parameter θ^{μ} for the policy function, that is, decide the optimal action for a given state. A critic is used for evaluating the policy function

estimated by the actor according to the temporal TD error (see Figure 9). One issue for DDPG is that it rarely explores actions. A solution is to add noise to the parameter space or the action space. It is claimed that adding to parameter space is better than to action space [39]. One commonly used noise is the Ornstein–Uhlenbeck random process. Algorithm 1 shows the pseudo-code of the proposed DDPG algorithm.

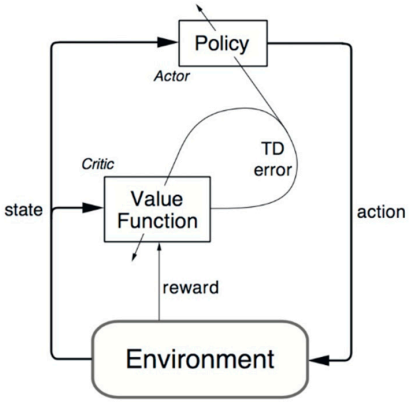

Figure 9. Actor–critic architecture.

Algorithm 1: Pseudo-code of the Deep deterministic policy gradient (DDPG) algorithm.

Randomly initialize critic network $Q(\mathrm{s},\mathrm{a}|\theta^{Q})$ and actor $\mu(\mathrm{s}|\theta^{\mu})$ with weights θ^{Q} and θ^{μ}.
Initialize target network Q' and μ' with weights $\theta^{Q'} \leftarrow \theta^{Q}$, $\theta^{\mu'} \leftarrow \theta^{\mu}$
Initialize replay buffer R
for episode = 1, M do
  Initialize a random process N for action exploration
  Receive initial observation state s_1
  for t = 1, T do
    Select action $a_t = \mu(s_t|\theta^{\mu}) + N_t$ according to the current policy and exploration noise
    Execute action a_t and observe reward r_t and observe new state s_{t+1}
    Store transition (s_t, a_t, r_t, s_{t+1}) in R
    Sample a random minibatch of N transitions (s_t, a_t, r_t, s_{t+1}) from R
    Set $y_i = r_i + \gamma Q'(s_{i+1}, \mu'(s_{i+1}|\theta^{\mu'})|\theta^{Q'})$
    Update critic by minimizing the loss: $\mathrm{L} = \frac{1}{N}\sum_i (y_i - Q(s_i, a_i|\theta^{Q}))^2$
    Update the actor policy using the sampled gradient:

$$\nabla_{\theta^{\mu}} \mathrm{J} \approx \frac{1}{N}\sum_{i} \nabla_{\mathrm{a}} \mathrm{Q}(\mathrm{s}, \mathrm{a}|\theta^{Q})|_{a=s_i, a=\mu(s_i)} \cdot \nabla_{\theta^{\mu}} \mu(\mathrm{s}|\theta^{\mu})|_{s_i}$$

    Update the target networks:

$$\theta^{Q'} \leftarrow \tau\theta^{Q} + (1-\tau)\theta^{Q'}$$
$$\theta^{\mu'} \leftarrow \tau\theta^{\mu} + (1-\tau)\theta^{\mu'}$$

  end for
end for

TensorFlow is one of the widely used end-to-end open-source platforms for machine learning. In order to draw on the research findings of DRL in other research fields, especially to re-use the existing program code frameworks in machine learning, we used Python compatible with TensorFlow as the algorithm design language in this study. Meanwhile, in order to apply the DRL algorithm built in Python to the diesel engine environment, we proposed to use MATLAB/Simulink as the program interface, so that the two-way transmission among Python, MATLAB/Simulink, and GT-SUITE could be reached. The specific DDPG algorithm implementation and the corresponding co-simulation platform

are shown in Figure 10. Key concepts of the DDPG-based boost control algorithm are formulated as follows.

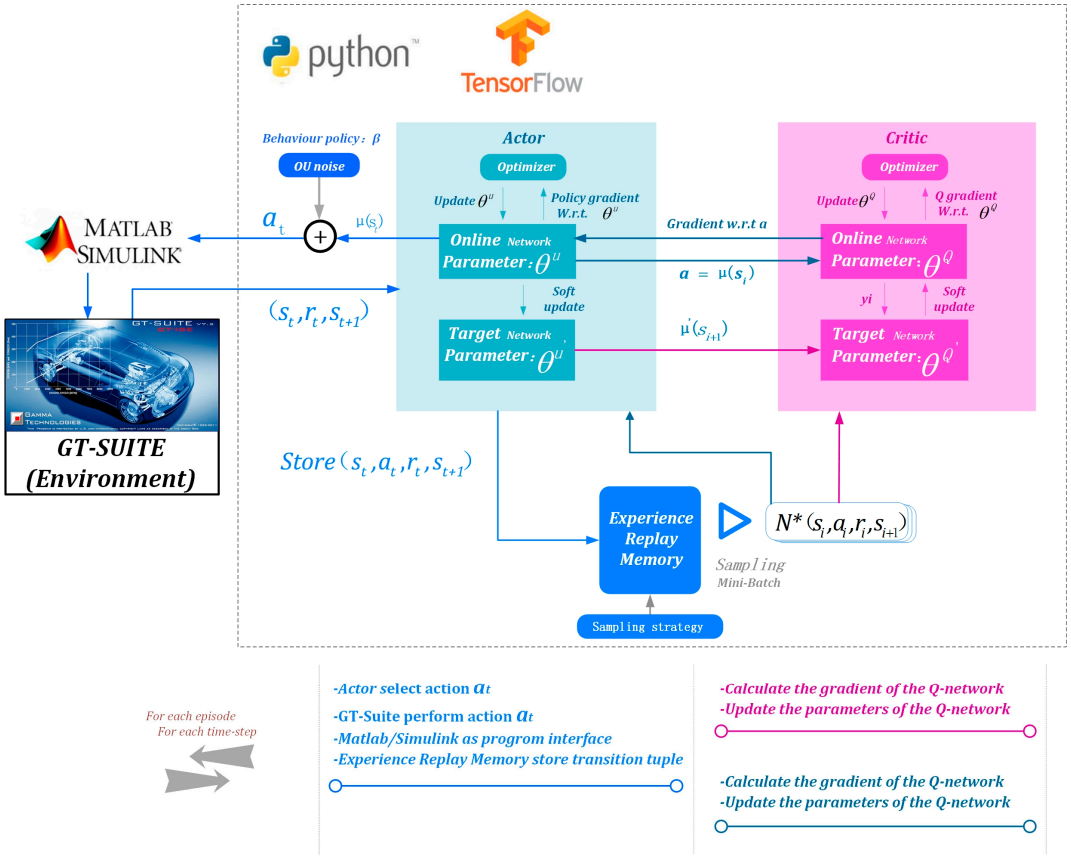

Figure 10. Deep deterministic policy gradient (DDPG) algorithm implementation and the corresponding co-simulation platform.

The engine speed, the actual boost pressure, the target boost pressure, and the current vane position were chosen to group a four-dimensional state space. It should be noted here that only a small number of states were chosen in this study in order to (a) facilitate the training process and (b) showcase the generalization ability of the DRL techniques. The vane position controlled by a membrane vacuum actuator was selected as the control action. Immediate reward is important in the RL algorithm because it directly affects the convergence curves and, in some cases, a fine adjustment of the immediate reward parameter can bring the final policy to the opposite poles. The agents always try to maximize the reward they can get by taking the optimal action at each time step because more cumulative rewards represent better overall control behavior. Therefore, the immediate reward should be defined based on optimization goals. The control objective of this work was to track the target boost pressure under transient driving cycles by regulating the vanes in a quick and stable manner. Keeping this objective in mind, the function of the error between the target and the current boost pressure and the rate of action change were defined as the immediate reward. The equation for the immediate reward is given as follows:

$$r_t = e^{-\frac{[0.95*|e(t)|+0.05*|I_t|]^2}{2}} - 1, \quad (6)$$

where r_t is the immediate reward generated when the state changes by taking an action at time t. $e(t)$ and I_t represent the error between the target and the current boost pressure and the rate of action change, respectively.

The corresponding DDPG parameters and the illustration of the actor–critic network are shown in Table 1 and Figure 11. In this study, the input layer of the actor network has four neurons, namely,

the engine speed, the actual boost pressure, the target boost pressure, and the current vane position. There are three hidden layers each having 120 neurons. The output layer has one neuron representing the control action (i.e., the vane position). All these layers are fully connected. For the critic network, the input layer has an additional neuron, which is the control action implemented by the actor network, compared to that of the critic network. There is one hidden layer having 120 neurons. The output layer of the critic network has one neuron representing the value function of the selected action for the specific state. The network is trained for 50 episodes and each episode represents the first 80% time of the FTP-72 trips (1098s).

Table 1. DDPG parameters.

Parameters	Value
Learning rate for actor	0.0001
Learning rate for critic	0.0002
Reward discount factor γ	0.9
Soft replacement factor τ	0.01
Replay memory size	100,000
Mini-batch size	256
Action randomness decay	0.999995
Initial exploration	10

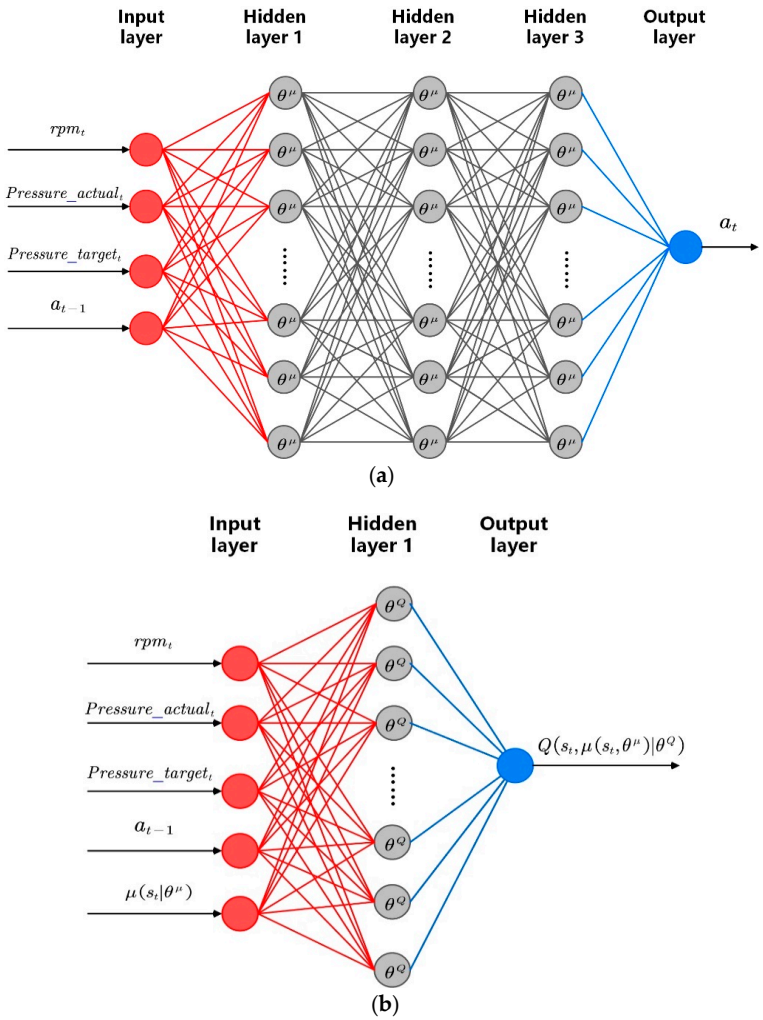

Figure 11. Illustration of the (**a**) actor network and (**b**) critic network.

4. Results and Discussion

In this article, the simulations were conducted based on an advanced co-simulation platform (see Figure 10). In order to validate its performance, the proposed DDPG algorithm was compared to a fine-tuned gain scheduled PID controller with both its P and I gains mapped as a function of speed and requested load. Without derivative action, a PI-controlled system may be less responsive, but it makes the system steadier at steady-state conditions (thus often adopted for industrial practice). It should be noted here that this PID controller adopted classic Ziegler–Nichols methods [40] to manually tune the control parameters, which took much effort and thus should be interpreted to represent a good control behavior benchmark. The US FTP-72 (Federal Test Procedure) driving cycle shown in Figure 7 was employed to verify the proposed strategy. The cycle simulated an urban route of 12.07 km with frequent stops and rapid accelerations. The maximum speed was 91.25 km/h and the average speed was 31.5 km/h. This transient driving cycle was selected because it mimics the real-world VGT environment system with large lag, strong coupling (especially with EGR) and nonlinear characteristics and thus, if a well-behaved control strategy in this environment is established, it should perform well in other driving cycles with more steady-state regions (such as the New European Driving Cycle (NEDC)). In this study, the first 80% time driving cycle was used to train the DDPG algorithm and the remaining data were destined for testing analysis. There are many different measures that can be used to compare the quality of controlled response. Integral absolute error (IAE), which integrates the absolute error over time, was used in this study to measure the control performance between the PID controller and the proposed DDPG algorithm.

Figure 12 shows the control performance using the fine-tuned PID controller. It can be seen that the actual boost pressure tracks the target boost pressure well at first glance. However, after zooming in on some operating conditions, a relatively large error can still be seen. This may be due to the turbo-lag, which cannot be improved from the control point of view (such as the time period from 10 s to 40 s, where although the VGT is already controlled to its minimum flow area for the fastest transient performance, it still exhibits lack of boost). Nevertheless, for most situations, taking the time period of 920 s to 945 s for example, there is still some room for the PID control strategy to improve. We note here that the results in Figure 12 are only a balance between control performance and tuning efforts, that is, a better control behavior can be achieved if the tuning process is made in a more finely manner, but more efforts and resources are required. In this research, the emphasis was not put on the final control performance comparison between PID and DRL theory, due to the fact that the structure behind each method is different and the control behavior, to a large extent, depends on how the control parameters are tuned. More focus was put on trying to solve the control problem in a self-learning manner and showcase good control adaptivity for the DRL approach.

The learning process of the DDPG algorithm can be seen in Figure 13. At the beginning of the learning, the cumulative rewards for the DDPG agent per episode were extremely low because (1) the agent (corresponding to the vane position actuator in the VGT boosting controller) only randomly selects actions in order to complete an extensive search process so as not to fall into local optimum and (2) the agent has no prior experience of what it should do for a specific state (thus the agent can only select the actions based on the initial DDPG parameters). After approximately 40 episodes, the algorithm has already been converged with the cumulative rewards, reaching a high value. This indicates that the agent has learned the experience to control the boost pressure. It should be noted that the learning process takes place only by direct interaction with the environment (in this case, the simulation software serves as the environment), without relying on model supervision or complete environment models, and a well-behaved control strategy is developed and finally formed autonomously from scratch. To answer the question of whether the learned controller was good enough, the control performance of the first 80% FTP-72 driving cycle using the final DDPG controller was compared with the performance based on the aforementioned fine-tuned PID controller. In Figures 12 and 14, it can be seen that both the PID and the proposed DDPG algorithm perform well at first glance, but after zooming in on some operating conditions, a large tracking disparity can still be seen. Although the PID controller seems to

track the boost pressure with relatively small errors, the control performance based on the proposed DDPG algorithm outperforms that of the PID controller with almost excellent tracking behavior. The IAE value of the PID control performance is 41.72, whereas the value based on the proposed DDPG algorithm can be as low as 37.43.

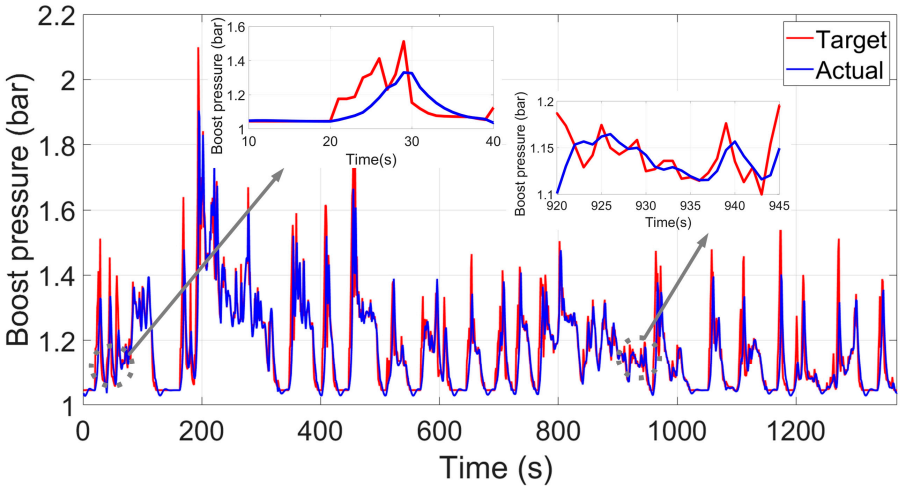

Figure 12. The fine-tuned gain-scheduled PID control performance.

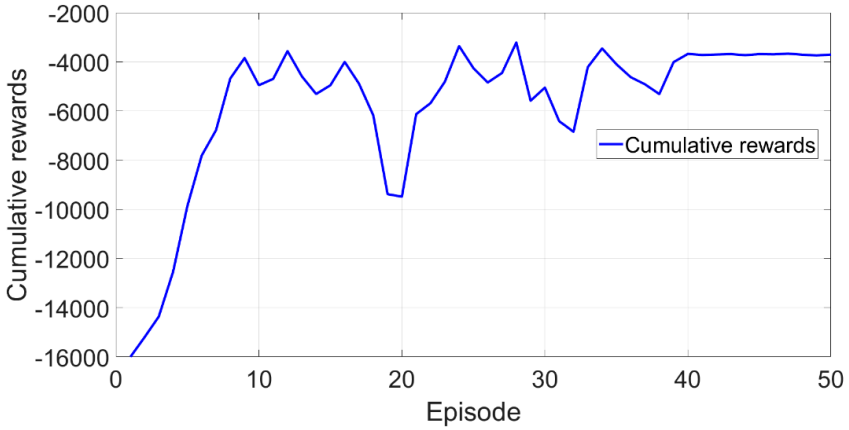

Figure 13. Track of all rewards per episode.

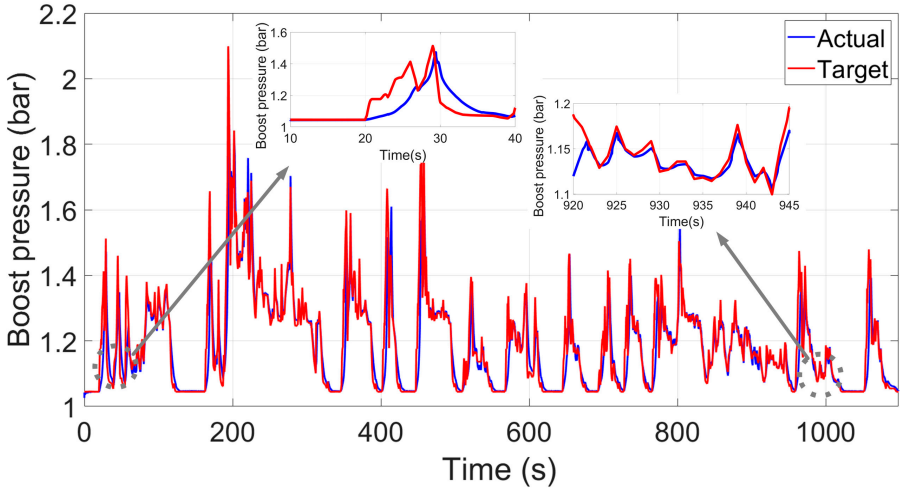

Figure 14. Control performance for the first 80% FTP-72 driving cycle using the final DDPG training parameters.

This difference is shown better in Figure 15, where the control performance comparison between the fine-tuned PID and the proposed DDPG from the time period of 920 s to 945 s is made. This time period was selected because it corresponds to the frequently used engine operating regions.

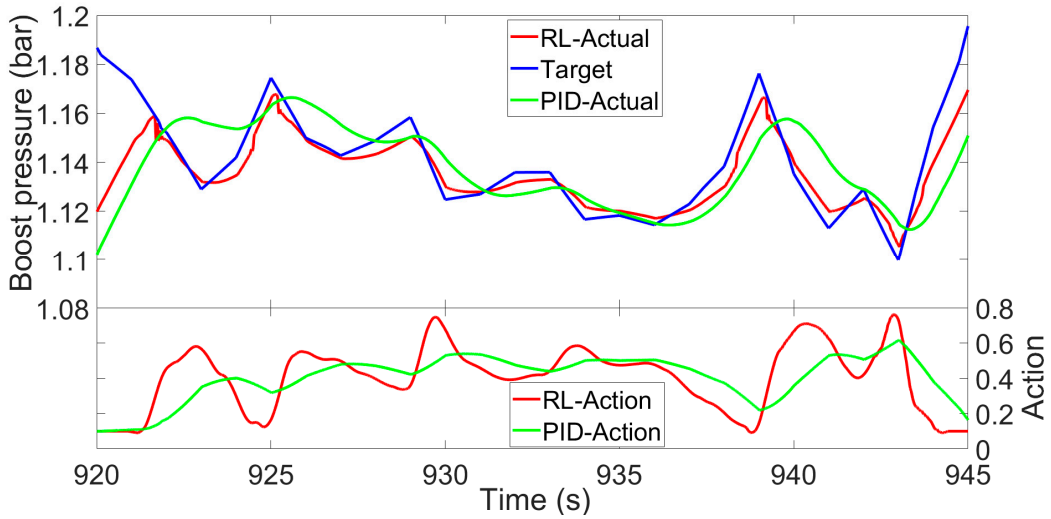

Figure 15. Control performance comparison between the fine-tuned PID and the proposed DDPG from the time period of 920 s to 945 s.

In order to showcase the generalization ability of the proposed DRL techniques, the control performance for the end 20% FTP-72 driving cycle based on the trained DDPG parameters was simulated. It can be seen in Figure 16 that although the control parameters were not trained based on this part of the driving cycle (i.e., some of the states may not have been visited in the previous training process), the performance still exhibits good control behavior. Compared to the same time period using the fine-tuned PID controller (already optimized for this time period), the control performance based on the proposed DDPG clearly performs better and the IAE of the PID and the proposed DDPG are 10.17 and 8.35, respectively.

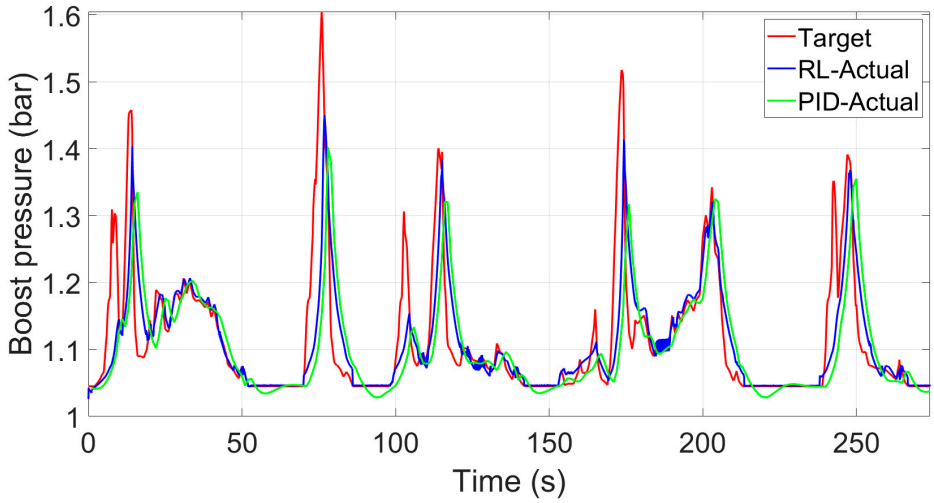

Figure 16. Control performance for the end 20% FTP-72 driving cycle based on the trained DDPG algorithm.

As the control strategy based on the proposed DDPG algorithm is able to achieve (or improve, depending on the tuning efforts of each algorithm) the control performance compared to a fine-tuned PID benchmark controller, it could replace the traditional PID controller for boosting control in the

near future. Compared to the benchmark PID controller whose parameters traditionally require manual adjustment (thus the tuning efficiency is low), the control strategy based on DDPG is able to adaptively adjust the algorithm strategy in the learning process, which not only can save a lot of manpower resources, but also adapt more to the changing environment and hardware aging over time (thus being unbiased by modelling errors). To prove this, another simulation model with a different combustion and turbocharger model was used. This was a simplified replication of a real engine plant whose transient performance could be diverged from the simulation prediction mainly due to combustion and turbocharger modelling inaccuracy. Figure 17 shows the control performance using both the pre-trained algorithm (which indicates the off-line behavior) and the strategy after continuing on-line learning in the "real engine" simulation model. It can be seen that the off-line policy is able to achieve a relatively good control behavior and can be improved further by continuing its learning from the interaction with the new environment on-line. Thus, different from other studies whose control parameters optimized in the simulation platform, for most cases, are no longer valid in the experimental test, the control strategy based on the proposed DRL techniques can combine the simulation training and the experimental continuing training together in order to fully utilize the computational resources off-line and refine the algorithm in the experimental environment on-line.

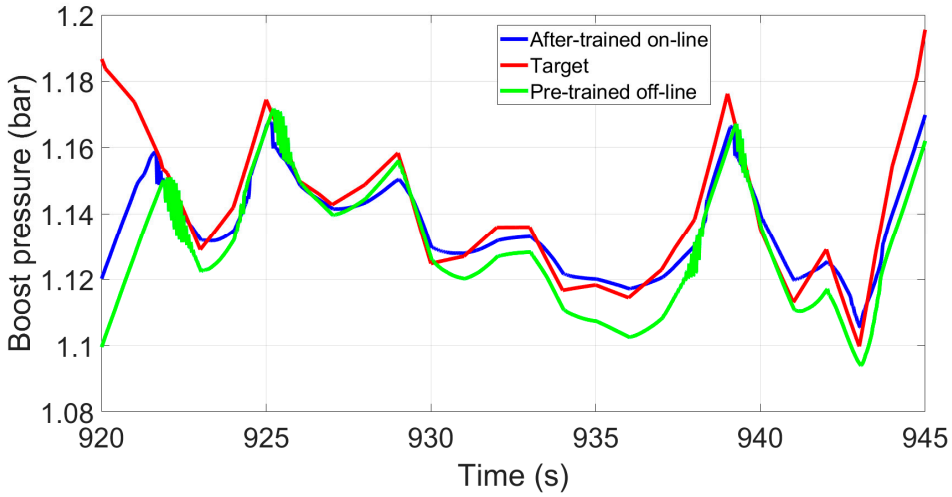

Figure 17. Control performance using the pre-trained off-line algorithm and the strategy after continuing on-line learning in the "real engine" simulation model.

Furthermore, because the learning process of the proposed DDPG algorithm distinguishes itself from other approaches by putting its emphasis on interacting with the transient environment, the final control performance is able to outperform that of the other approaches whose techniques are only based on the steady-state simulation or experimental control behavior. The most obvious example would be its capability to exceed the classic feedforward control which only responds to its control signal in a pre-defined way without responding to how the load reacts. It is known that most of the pre-defined map in a controller with feedforward function is calibrated in a steady-state environment in industry and is fixed for the entire product lifecycle. The proposed DDPG algorithm, however, because the control action adapts to the environment, is equivalent to the concept of the so-called automatic transient calibration.

5. Conclusions and Future Work

In this paper, a model-free self-learning DDPG algorithm was employed for the boost control of a VGT-equipped engine with the aim to compare the DDPG techniques with traditional control methods and to provide references for the further development of DDPG algorithms on sequential decision-control problems with other industrial applications. Using a fine-tuned PID controller as

a benchmark, the results show that the control performance based on the proposed DDPG algorithm can achieve a good transient control performance from scratch by autonomously learning the interaction with the environment, without relying on model supervision or complete environment models. In addition, the proposed strategy is able to adapt to the changing environment and hardware aging over time by adaptively tuning the algorithm in a self-learning manner on-line, making it attractive to real plant control problems whose system consistency may not be strictly guaranteed and whose environment may change over time. This indicates that the control strategy based on the proposed DRL techniques can combine the simulation training and the experimental continuing training together in order to fully utilize the computational resources off-line and refine the algorithm in the experimental environment on-line. Future work may include applying DRL-based parallel computer architecture to boost the computational efficiency for the control problem with high-order, large lag, strong coupling, and nonlinear characteristics. Another interesting direction could be combining some look-ahead strategies with the proposed DRL techniques to accelerate the training process and improve the final control performance. The stability control between multiple reinforcement learning-based controllers will also be studied, which includes distributed RL and hierarchical RL.

Author Contributions: B.H. drafted the paper and provided the overall research ideas. J.Y. provided reinforcement learning strategy suggestions. S.L. and J.L. analyzed the data. H.B. revised the paper. These authors are contributed equally to this work.

Funding: This work was supported by the National Natural Science Foundation of China (Grant No. 51905061), the Chongqing Natural Science Foundation (Grant No. cstc2019jcyj-msxmX0097), the Science and Technology Research Program of Chongqing Municipal Education Commission (Grant No. KJQN201801124), the Venture & Innovation Support Program for Chongqing Overseas Returners (Grant No. cx2018135), and the Open Project Program of the State Key Laboratory of Engines (Tianjin University) (Grant No. k2019-02), to whom the authors wish to express their thanks.

Acknowledgments: The authors would like to thank the anonymous reviewers for their valuable comments and suggestions.

Conflicts of Interest: The authors declare no conflict of interest.

Data Availability: The data used to support the findings of this study are available from the corresponding author upon request.

Nomenclature

AC	Actor critic
BMEP	Brake mean effective pressure
DQN	Deep Q-network
DDPG	Deep deterministic policy gradient
DI	Direct injection
DP	Dynamic programming
DRL	Deep reinforcement learning
DPF	Diesel particulate filter
DOC	Diesel oxidation catalyst
EGR	Exhaust gas recirculation
FMEP	Friction mean effective pressure
FTP	Federal Test Procedure
FTP-72	Federal Test Procedure-72
HIL	Hardware-in-the-loop
IAE	Integral absolute error
FMEP	Friction mean effective pressure
IMEP	Indicated mean effective pressure
MDP	Markov decision process
MVEM	Mean value engine model
NEDC	New European Driving Cycle
PID	Proportion integration differentiation

References

1. Awad, O.I.; Ali, O.M.; Mamata, R.; Abdullaha, A.A.; Najafid, G.; Kamarulzamana, M.K.; Yusria, I.M.; Noora, M.M. Using fusel oil as a blend in gasoline to improve si engine efficiencies: A comprehensive review. *Renew. Sustain. Energy Rev.* **2017**, *69*, 1232–1242. [CrossRef]
2. Enang, W.; Bannister, C. Modelling and control of hybrid electric vehicles (A comprehensive review). *Renew. Sustain. Energy Rev.* **2017**, *74*, 1210–1239. [CrossRef]
3. Hu, B.; Akehurst, S.; Brace, C. Novel approaches to improve the gas exchange process of downsized turbocharged spark-ignition engines: A review. *Int. J. Engine Res.* **2015**, *17*, 1–24. [CrossRef]
4. Hu, B.; Turner, J.W.G.; Akehurst, S.; Brace, C.J.; Copeland, C. Observations on and potential trends for mechanically supercharging a downsized passenger car engine: A review. *Proc. Inst. Mech. Eng. Part D J. Automob. Eng.* **2016**, *231*, 435–456. [CrossRef]
5. Hu, B.; Akehurst, S.; Lewis, A.G.J.; Lu, P.; Millwood, D.; Copeland, C.; Chappell, E.; Freitas, A.D.; Shawe, J.; Burtt, D. Experimental analysis of the V-Charge variable drive supercharger system on a 1.0 L GTDI engine. *Proc. Inst. Mech. Eng. Part D J. Automob. Eng.* **2017**, *232*, 449–465. [CrossRef]
6. Tang, H.; Pennycott, A.; Akehurst, S.; Brace, C.J. A review of the application of variable geometry turbines to the downsized gasoline engine. *Int. J. Engine Res.* **2015**, *16*, 810–825. [CrossRef]
7. Zhao, D.; Winward, E.; Yang, Z.; Stobart, R.; Mason, B.; Steffen, T. An integrated framework on characterization, control, and testing of an electrical turbocharger assist. *IEEE Trans. Ind. Electron.* **2018**, *65*, 4897–4908. [CrossRef]
8. Oh, B.; Lee, M.; Park, Y.; Won, J.; Sunwoo, M. Mass air flow control of common-rail diesel engines using an artificial neural network. *Proc. Inst. Mech. Eng. Part D J. Automob. Eng.* **2013**, *227*, 299–310. [CrossRef]
9. Xu, S.; Hashimoto, S.; Jiang, W.; Jiang, Y.; Izaki, K.; Kihara, T.; Ikeda, R. Slow Mode-Based Control Method for Multi-Point Temperature Control System. *Processes* **2019**, *7*, 533. [CrossRef]
10. Xu, S.; Hashimoto, S.; Jiang, W. Pole-Zero Cancellation Method for Multi Input Multi Output (MIMO) Temperature Control in Heating Process System. *Processes* **2019**, *7*, 497. [CrossRef]
11. Sekour, M.; Hartani, K.; Draou, A.; Ahmed, A. Sensorless Fuzzy Direct Torque Control for High Performance Electric Vehicle with Four In-Wheel Motors. *J. Electr. Eng. Technol.* **2013**, *8*, 530–543. [CrossRef]
12. Zhang, X.; Sun, L.; Zhao, K.; Sun, L. Nonlinear Speed Control for PMSM System Using Sliding-Mode Control and Disturbance Compensation Techniques. *IEEE Trans. Power Electr.* **2012**, *28*, 1358–1365. [CrossRef]
13. Gao, R.; Gao, Z. Pitch control for wind turbine systems using optimization. estimation and compensation. *Renew. Energy* **2016**, *91*, 501–515. [CrossRef]
14. Chee, F.; Fernando, T.L.; Savkin, A.V.; Heeden, V.V. Expert PID Control System for Blood Glucose Control in Critically Ill Patients. *IEEE Trans. Inf. Technol. Biomed.* **2003**, *7*, 419–425. [CrossRef] [PubMed]
15. Savran, A. A multivariable predictive fuzzy PID control system. *Appl. Soft Comput.* **2013**, *13*, 2658–2667. [CrossRef]
16. Lopez-Franco, C.; Gomez-Avila, J.; Alanis, A.Y.; Arana-Daniel, N.; Villaseñor, C. Visual Servoing for an Autonomous Hexarotor Using a Neural Network Based PID Controller. *Sensors* **2017**, *17*, 1865. [CrossRef] [PubMed]
17. Omran, R.; Younes, R.; Champoussin, J. Neural networks for real-time nonlinear control of a variable geometry turbocharged diesel engine. *Int. J. Robust Nonlin.* **2008**, *18*, 1209–1229. [CrossRef]
18. Wahlstrom, J.; Eriksson, L.; Nielsen, L. EGR-VGT Control and Tuning for Pumping Work Minimization and Emission Control. *IEEE Trans. Contr. Syst. Technol.* **2010**, *18*, 993–1003. [CrossRef]
19. Zamboni, G.; Capobianco, M. Influence of high and low pressure EGR and VGT control on in-cylinder pressure diagrams and rate of heat release in an automotive turbocharged diesel engine. *Appl. Therm. Eng.* **2013**, *51*, 586–596. [CrossRef]
20. Sutton, R.S.; Barto, A.G. *Reinforcement Learning: An Introduction*, 2nd ed.; The MIT Press: Cambridge, MA, USA, 2018; pp. 97–113.
21. Gao, Z.; Saxen, H.; Gao, C. Special section on data-driven approaches for complex industrial systems. *IEEE Trans. Ind. Inform.* **2013**, *9*, 2210–2212. [CrossRef]
22. Gao, Z.; Nguang, S.K.; Kong, D.X. Advances in Modelling, Monitoring, and Control for Complex Industrial Systems. *Complexity* **2019**. [CrossRef]
23. Sutton, R.S. Learning to predict by the methods of temporal differences. *Mach. Learn.* **1988**, *3*, 9–44. [CrossRef]

24. Liu, T.; Zou, Y.; Liu, D.; Sun, F. Reinforcement Learning–Based Energy Management Strategy for a Hybrid Electric Tracked Vehicle. *Energies* **2015**, *8*, 7243–7260. [CrossRef]
25. Cao, H.; Yu, T.; Zhang, X.; Yang, B.; Wu, Y. Reactive Power Optimization of Large-Scale Power Systems: A Transfer Bees Optimizer Application. *Processes* **2019**, *7*, 321. [CrossRef]
26. Francois-Lavet, V.; Henderson, P.; Islam, R.; Bellemare, M.G.; Pineau, J. An introduction to deep reinforcement learning. *Found. Trends Mach. Learn.* **2018**, *11*, 245–246. [CrossRef]
27. Silver, D.; Aja, H.; Maddison, J.C.; Guez, A.; Sifre, L.; van den Driessche, G.; Schrittwieser, J.; Antonoglou, L.; Panneershelvam, V.; Lanctot, M.; et al. Mastering the game of go with deep neural networks and tree search. *Nature* **2016**, *529*, 484–489. [CrossRef]
28. Silver, D.; Schrittwieser, J.; Simonyan, K.; Antonoglou, I.; Huang, A.; Guez, A.; Hubert, T.; Baker, L.; Lai, M.; Bolton, A.; et al. Mastering the Game of Go without Human Knowledge. *Nature* **2017**, *550*, 354–359. [CrossRef]
29. Mnih, V.; Kavukcuoglu, K.; Silver, D.; Graves, A.; Antonoglou, I.; Wierstra, D.; Riedmiller, M. Playing Atari with Deep Reinforcement Learning. In Proceedings of the NIPS Deep Learning Workshop 2013, Lake Tahoe, NV, USA, 9 December 2013.
30. Lillicrap, T.P.; Hunt, J.J.; Pritzel, A.; Heess, N.; Erez, T.; Tassa, Y.; Silver, D.; Wierstra, D. Continuous control with deep reinforcement learning. In Proceedings of the International Conference on Learning Representations 2016, San Juan, PR, USA, 2–4 May 2016.
31. Silver, D.; Level, G.; Heess, N.; Degris, T.; Wierstra, D.; Riedmiller, M. Deterministic Policy Gradient Algorithms. In Proceedings of the 31st International Conference on International Conference on Machine Learning 2014, Beijing, China, 21–26 June 2014.
32. Nikzadfar, K.; Shamekhi, A.H. An extended mean value model (EMVM) for control-oriented modeling of diesel engines transient performance and emissions. *Fuel* **2015**, *154*, 275–292. [CrossRef]
33. Jiang, M.; Jin, Q. Multivariable System Identification Method Based on Continuous Action Reinforcement Learning Automata. *Processes* **2019**, *7*, 546. [CrossRef]
34. Zhang, D.; Gao, Z. Reinforcement learning–based fault-tolerant control with application to flux cored wire system. *Meas. Control* **2018**, *51*, 349–359. [CrossRef]
35. Kulkarni, T.D.; Narasimhan, K.R.; Saeedi, A.; Tenenbaum, J. Hierarchical Deep Reinforcement Learning: Integrating Temporal Abstraction and Intrinsic Motivation. In Proceedings of the 30th Conference on Neural Information Processing Systems 2016, Barcelona, Spain, 5–10 December 2016.
36. Howard, R.A. *Dynamic Programming and Markov Process*; The MIT Press: Cambridge, MA, USA, 1966.
37. Wang, Y.; Velswamy, K.; Huang, B. A Long-Short Term Memory Recurrent Neural Network Based Reinforcement Learning Controller for Office Heating Ventilation and Air Conditioning Systems. *Processes* **2017**, *5*, 46. [CrossRef]
38. Zhang, D.; Lin, Z.; Gao, Z. A Novel Fault Detection with Minimizing the Noise-Signal Ratio Using Reinforcement Learning. *Sensors* **2018**, *18*, 3087. [CrossRef] [PubMed]
39. Silver, D.; Lever, G. Better Exploration with Parameter Noise. Available online: https://openai.com/blog/better-exploration-with-parameter-noise/ (accessed on 28 May 2019).
40. ÅströmK, J.; Hägglund, T. *PID Controllers: Theory, Design, and Tuning*; Instrument Society of America: Research Triangle Park, NC, USA, 1995; Volume 2.