Frontiers of Information Technology & Electronic Engineering
www.jzus.zju.edu.cn; engineering.cae.cn; www.springerlink.com
ISSN 2095-9184 (print); ISSN 2095-9230 (online)
E-mail: jzus@zju.edu.cn

Pre-training with asynchronous supervised learning for reinforcement learning based autonomous driving*

Yunpeng WANG†, Kunxian ZHENG†‡, Daxin TIAN†, Xuting DUAN†, Jianshan ZHOU

Beijing Advanced Innovation Center for Big Data and Brain Computing,
School of Transportation Science and Engineering, Beihang University, Beijing 100191, China

†E-mail: ypwang@buaa.edu.cn; zhengkunxian@buaa.edu.cn; dtian@buaa.edu.cn; duanxuting@buaa.edu.cn

Received Nov. 20, 2019; Revision accepted Dec. 29, 2020; Crosschecked Feb. 3, 2021

Abstract: Rule-based autonomous driving systems may suffer from increased complexity with large-scale intercoupled rules, so many researchers are exploring learning-based approaches. Reinforcement learning (RL) has been applied in designing autonomous driving systems because of its outstanding performance on a wide variety of sequential control problems. However, poor initial performance is a major challenge to the practical implementation of an RL-based autonomous driving system. RL training requires extensive training data before the model achieves reasonable performance, making an RL-based model inapplicable in a real-world setting, particularly when data are expensive. We propose an asynchronous supervised learning (ASL) method for the RL-based end-to-end autonomous driving model to address the problem of poor initial performance before training this RL-based model in real-world settings. Specifically, prior knowledge is introduced in the ASL pre-training stage by asynchronously executing multiple supervised learning processes in parallel, on multiple driving demonstration data sets. After pre-training, the model is deployed on a real vehicle to be further trained by RL to adapt to the real environment and continuously break the performance limit. The presented pre-training method is evaluated on the race car simulator, TORCS (The Open Racing Car Simulator), to verify that it can be sufficiently reliable in improving the initial performance and convergence speed of an end-to-end autonomous driving model in the RL training stage. In addition, a real-vehicle verification system is built to verify the feasibility of the proposed pre-training method in a real-vehicle deployment. Simulations results show that using some demonstrations during a supervised pre-training stage allows significant improvements in initial performance and convergence speed in the RL training stage.

Key words: Self-driving; Autonomous vehicles; Reinforcement learning; Supervised learning
https://doi.org/10.1631/FITEE.1900637 **CLC number:** TP181; U495

1 Introduction

As an important component of intelligent transportation systems, autonomous driving has the potential to increase the safety and efficiency of future transportation systems. By carefully considering the autonomous driving problem, it can be mathematically mapped into a sequential decision-making problem embedded in complicated time-varying environments. Hence, with the goal of solving this issue, many researchers resort to the reinforcement learning (RL) based theory. As we can see, the emerging RL method shows great prospects in sequential decision-making problems (Mnih et al., 2013, 2015, 2016; Li L et al., 2016; Silver et al., 2017; Wang et al., 2020), and a variety of RL-based solutions have been applied in dynamic control and other domains (Mao et al., 2016; Liu et al., 2017; Xu et al., 2017;

‡ Corresponding author

* Project supported by the National Natural Science Foundation of China (Nos. 61672082 and 61822101), the Beijing Municipal Natural Science Foundation, China (No. 4181002), and the Beihang University Innovation and Practice Fund for Graduate, China (No. YCSJ-02-2018-05)

ORCID: Kunxian ZHENG, https://orcid.org/0000-0002-2887-9294

© Zhejiang University Press 2021

He Y et al., 2018). The extensive success of deep RL has generated many more applications using deep RL for end-to-end autonomous driving model's training tasks (Sallab et al., 2017; Bai et al., 2019; Chen JY et al., 2019). One of the most impressive accomplishments of deep RL is its ability to learn directly from raw images, and to achieve state-of-the-art results with little prior human knowledge. However, exactly because of this lack of prior knowledge, deep RL usually suffers from poor initial performance (Brys et al., 2015; de la Cruz et al., 2019), so it requires extensive training time (too many real-world experience data) before being practically used in autonomous driving tasks. This defect can be tolerated in a simulation environment, such as when training the racing game robots; however, outside the simulation environment, things are not as simple. A long training time means more human and financial investment, and for autonomous driving, it also brings greater risks.

Using humans for demonstrations is one method of speeding up the training process of a deep RL-based model. In this study, we introduce prior knowledge into an end-to-end autonomous driving model by supervised learning (SL) training to address the problem of poor initial performance before RL training in real-world settings. There have been some approaches proposed to combine deep RL with prior knowledge, from reward shaping (Brys et al., 2015; Nair et al., 2018) to feature extraction pre-training (de la Cruz et al., 2019). The reward-shaping approach may not work well in the continuous reward space and it still does not address the problem of poor initial performance. The feature extraction pre-training approach divides the deep RL-based model into two parts, namely, the feature extractor and the controller, and pre-trains only the convolutional layers as a feature extractor. However, Nvidia's research (Bojarski et al., 2016) shows that it is not possible to make a clean break between the part of the end-to-end network that functions primarily as a feature extractor and the part that serves as a controller. Therefore, the feature extraction pre-training approach may interfere with subsequent RL training processes in real-world settings. In RL-based end-to-end autonomous driving model's pre-training tasks, the pre-training algorithms should consider the integrality of the model and the increased cost with large-scale demonstrations. The difficulty lies in accurately modeling this temporal correlation characteristic and making better driving decisions, to maximize system performance in a long-term dynamic vehicle mobility environment period. The limitation of existing works and the challenges in RL-based model's pre-training tasks as previously mentioned, motivate us to explore a novel methodological framework.

The pre-training problem can be mathematically mapped into an imitation learning problem embedded in a sequential decision-making process. In this work, we propose an asynchronous supervised learning (ASL) method for pre-training the end-to-end autonomous driving model, similar to the Gorila framework (Nair et al., 2015) and the asynchronous RL framework (Mnih et al., 2016); however, instead of executing multiple RL processes, we asynchronously execute multiple SL processes in parallel, on multiple real-world training data sets. The aim in designing this asynchronous method is to propose an SL algorithm that can learn a sequential decision-making policy reliably and without large resource requirements. By running different SL processes in different threads, the overall changes made to the model parameters by multiple agents applying online updates in parallel are likely to be less correlated in time than a single agent applying online updates, which stabilizes the SL process. After pre-training by the ASL method, the autonomous driving model is trained in real-world settings by RL. In fact, combining the proposed pre-training method with RL constitutes a new joint training framework. Pre-training introduces prior knowledge to address the problem of poor initial performance; RL then breaks the performance limit, and achieves performance beyond humans. To avoid collecting expensive human driving demonstration data, a manually designed heuristic driving policy (MDHDP) is used to generate high-reward experiences as demonstrations.

The main contributions of this paper are summarized as follows:

1. Although the existing RL theory has achieved some success in a variety of domains, its applicability has previously been focused on gaming or robotic control domains or on other domains in which poor initial performance can be tolerated. Few efforts have been made to explore the applicability of SL theory in promoting initial performance of RL-based models for practical applications in real-world

settings. We conduct some innovative work to demonstrate the effectiveness and the potential of the joint training framework of SL and RL in the design of the pre-training method, for enhancement of the initial performance of the RL-based model.

2. Moreover, while our study is based on the existing RL theory, the methodological framework proposed is not a direct application of existing RL algorithms. To facilitate the practical application of the RL-based autonomous driving model in real-world settings, we propose a novel pre-training approach, in which two new components are designed and combined to adapt to the environments and improve the initial decision-making performance of RL agents in autonomous driving problems: (1) an ASL method based on the joint methodological framework of RL and SL for the pre-training stage, and (2) an MDHDP for automatic collection of pre-training demonstration data.

3. It is challenging to implement visual analysis for a neural network that outputs continuous actions. We propose a visualization method based on the basic theory of single-variable analysis, to examine the improvements in the RL-based autonomous driving model after pre-training. The results of visualization are presented in the form of heatmaps, which directly show the specific areas that have important impacts on decision making. This kind of analysis can demonstrate the effectiveness of pre-training from a detailed point of view, which is significant in testing and validating end-to-end autonomous driving systems.

2 Related work

Conventional autonomous driving systems are manually designed (González et al., 2016; Paden et al., 2016; Hao et al., 2018; Schwarting et al., 2018), and driving actions are determined according to the rule base. However, this rule-based system structure is too bloated, so more and more researchers are investing in the study of learning-based approaches (Bojarski et al., 2016; Codevilla et al., 2018; Li LZ et al., 2018). The RL-based end-to-end model is an increasingly popular choice of autonomous driving. Bai et al. (2019) used an improved deep Q-network to learn high-level driving behavior policy from a customized data format called the hyper-grid matrix. However, the Q-learning-based model is not sufficient to be an end-to-end autonomous driving model, because Q-learning can output only discrete actions. Chen JY et al. (2019) used a bird-view representation as the input of a deep RL-based model, and then trained and evaluated it in the CARLA simulator. Both the bird-view representation and the hyper-grid matrix mentioned above have undergone a series of processing steps, such as data extraction, fusion, and transformation, which may cause a loss of some crucial information. In this work, our end-to-end autonomous driving model can directly take the raw sensor data from a front-facing camera as continuous driving action inputs and outputs, such as the wheel steering angle.

To run in real-world settings, a learning-based autonomous driving model has to be trained using real-world experiences. We must realize that the "trial-and-error course" used by the RL-based model is very dangerous for training an autonomous driving vehicle in real-world settings. Specifically, poor initial performance requires much more manual intervention in the RL training process to avoid accidents. Now there is some research into introducing prior knowledge in the training of RL-based models in the fields of games, robots, and so on. Brys et al. (2015) modified the reward function to drive an RL-based agent to explore states with high potential, but this method still suffers from poor initial performance. de la Cruz et al. (2019) pre-trained the convolutional layers of a deep RL-based model to prevent wasting time in learning feature extraction, but this method undermines the model's overall coordination. Zhang and Ma (2018) used a trained RL-based model to generate demonstration data to guide the training process of an untrained model. However, this work still does not address the problem of poor initial performance. In this study, we combine SL with RL to improve the initial performance of the RL-based model. We use MDHDP to drive a vehicle and generate demonstrations to form SL training data sets. Then, by the asynchronous multi-threading SL process, our RL-based model learns prior knowledge before RL training in real-world settings.

To analyze the improvement due to pre-training, we visualize the specific influences of different input units in the neural network. Selvaraju et al. (2019) proposed gradient-weighted class activation mapping (Grad-CAM) to visualize the feature maps in the last convolutional layer to analyze the

influence of different input units on the classification result. Compared to class activation mapping (CAM) (Zhou et al., 2016), Grad-CAM has the advantage that it does not need any modification of the network structure. The methods above are not adopted in this study because they are applicable only to models with discrete output, whereas our autonomous driving model outputs continuous results. Thus, we propose a visualization approach that is applicable to models with continuous output.

At present, there are few research reports on joint methodological framework based pre-training-oriented solutions in real-world settings. The research in this study fully demonstrates the effectiveness and advantages of the joint methodological framework of SL and RL in this field, and enriches the intellectual system of RL-based autonomous driving training tasks. This paper innovatively proposes an ASL method based on the joint methodological framework, and an MDHDP for automatic collection of demonstration data.

3 System model and problem formulation

The pre-training system is defined by the five-tuple array $\{\mathbb{S}, \mathbb{A}, \mathbb{L}, \mathbb{P}, \gamma\}$. In this array, $\mathbb{S}$ is a set of states, $\mathbb{A}$ is a set of actions, $\mathbb{L}$ is the loss function, $\mathbb{P}$ is the state transition probability, and $\gamma \in [0, 1]$ is the discount factor that is used to balance the weight between real-time loss and long-term loss. When $\gamma = 0$, the RL-based agent considers only real-time loss, and $\gamma = 1$ means that the long-term loss and the real-time loss are equally important. We map the pre-training problem into the formulation as follows:

1. State

Let $\mathbb{S}_i = \{\mathcal{S}_{i1}, \mathcal{S}_{i2}, \ldots, \mathcal{S}_{iN}\}$ be the set of states that autonomous driving vehicle i can directly observe by its front-facing camera, where $\mathcal{S}_{in}$ $(1 \le n \le N)$ represents the state of the n^{th} experience in the demonstration data set Ω_i. Specifically, $\mathcal{S}_{in}$ consists of four pre-processed front-facing camera images.

2. Action

$\mathcal{A}_{in} \in \mathbb{A}_i = \{\mathcal{A}_{i1}, \mathcal{A}_{i2}, \ldots, \mathcal{A}_{iN}\}$, where $\mathcal{A}_{in} \in [-1, 1]$ represents the steering of the n^{th} experience, "-1" for a full left turn and "1" for a full right turn.

3. Loss

We use $L_{in}(\mathcal{S}_{in}, \mathcal{A}_{in})$, $L_{in} \in \mathbb{L}_i = \{L_{i1}, L_{i2}, \ldots, L_{iN}\}$, to denote the loss of the n^{th} experience. We formulate the loss L_{in} by

$$L_{in}(\mathcal{S}_{in}, \mathcal{A}_{in}) = \varsigma - \varrho \|\mathcal{A}_{in} - \mathcal{A}'_{in}\|_1, \tag{1}$$

where ς and ϱ are variables that change with $\|\mathcal{A}_{in} - \mathcal{A}'_{in}\|_1$. $\mathcal{A}'_{in}$ denotes the action taken by our end-to-end autonomous driving model after inputting state $\mathcal{S}_{in}$, and $\mathcal{A}_{in}$ denotes the demonstration action of the n^{th} experience in Ω_i.

4. State transition probability

Given the current state $\mathcal{S}_{it}$ in the t^{th} time slot, the probability that $\mathcal{S}_{it}$ transits to the next time slot state $\mathcal{S}_{i(t+1)}$ after taking action $\mathcal{A}'_{it}$ is denoted by $p_{it}(\mathcal{S}_{i(t+1)} \mid \mathcal{S}_{it}, \mathcal{A}'_{it})$, where $\mathcal{S}_{i(t+1)} \in \mathbb{S}'_i = \{\mathcal{S}'_1, \mathcal{S}'_2, \ldots, \mathcal{S}'_U\}$, and $\mathbb{S}'_i$ is the set of all possible states in the $(t+1)^{\text{th}}$ time slot.

Given a stochastic policy $\pi_i(\mathcal{S}_{it})$ that outputs the probability distribution of action $\mathcal{A}'_{it}$ based on state $\mathcal{S}_{it}$, we use the state value function to represent the total loss that can be obtained by executing policy π_i to the final state $\mathcal{S}_{iT}$ from an initial state $\mathcal{S}_{it}$. We assume that the n^{th} experience is collected in the t^{th} time slot, so $\mathcal{S}_{it}$ and $\mathcal{S}_{in}$ are equivalent. Note that the state $\mathcal{S}_{i(n+1)}$ of the $(n+1)^{\text{th}}$ experience in Ω_i is not the next time slot state $\mathcal{S}_{i(t+1)}$ transited by $\mathcal{S}_{in}$ after agent i performs action $\mathcal{A}'_{in}$ (similarly, $\mathcal{A}'_{in}$ and $\mathcal{A}'_{it}$ are equivalent). In fact, after the demonstration data collection is completed, the order of data will be shuffled and rearranged, so $\mathcal{S}_{i(n+1)}$ is just behind $\mathcal{S}_{in}$, and there is no causal relationship between it and $\mathcal{S}_{in}$. We can do some image processing on state $\mathcal{S}_{in}$, such as zooming and rotating, to obtain the next state $\mathcal{S}_{i(t+1)}$ after action $\mathcal{A}'_{in}$ is executed. The state value function $V^{\pi_i}(\mathcal{S}_{it})$ is as follows:

$$\begin{aligned} V^{\pi_i}(\mathcal{S}_{it}) &= \mathbb{E}_{\mathcal{A}'_{it},\ldots\sim\pi_i}\mathbb{E}_{\mathcal{S}_{i(t+1)},\ldots\sim p_i}\left\{\sum_{\tau=0}^{T-t}\gamma^\tau L_{i(t+\tau)}\right\} \\ &= \mathbb{E}_{\mathcal{A}'_{it}\sim\pi_i}\mathbb{E}_{\mathcal{S}_{i(t+1)}\sim p_i}\{L_{it}(\mathcal{S}_{it}, \mathcal{A}'_{it}) + \gamma V^{\pi_i}(\mathcal{S}_{i(t+1)})\}. \end{aligned} \tag{2}$$

If the agent performs action $\mathcal{A}''_{it}$ without the policy π_i in a certain state $\mathcal{S}_{it}$, and then performs actions in accordance with policy π_i in subsequent states, the expected total loss $L^{\pi_i}(\mathcal{S}_{it}, \mathcal{A}''_{it})$ obtained by the agent is as follows:

$$\begin{aligned} &L^{\pi_i}(\mathcal{S}_{it}, \mathcal{A}''_{it}) \\ =&\mathbb{E}_{\mathcal{S}_{i(t+1)},\ldots\sim p_i}\mathbb{E}_{\mathcal{A}'_{i(t+1)},\ldots\sim\pi_i}\left\{\sum_{\tau=0}^{T-t}\gamma^\tau L_{i(t+\tau)}\right\} \\ =&\mathbb{E}_{\mathcal{S}_{i(t+1)}\sim p_i}\{L_{it}(\mathcal{S}_{it}, \mathcal{A}''_{it}) + \gamma V^{\pi_i}(\mathcal{S}_{i(t+1)})\}. \end{aligned} \tag{3}$$

Furthermore, the advantage function $A^{\pi_i}(\mathcal{S}_{it}, \mathcal{A}''_{it})$ is as follows:

$$A^{\pi_i}(\mathcal{S}_{it}, \mathcal{A}''_{it}) = L^{\pi_i}(\mathcal{S}_{it}, \mathcal{A}''_{it}) - V^{\pi_i}(\mathcal{S}_{it}), \quad (4)$$

where $A^{\pi_i}(\mathcal{S}_{it}, \mathcal{A}''_{it})$ represents the advantage due to selecting another action instead of the one determined by policy π_i. Given the certain state $\mathcal{S}_{it}$, our target is to find the best policy $\pi_i^*(\mathcal{S}_{it})$ to maximize Eq. (2) by using Eq. (4). That is, $\pi_i^*(\mathcal{S}_{it})$ satisfies the following expression:

$$\pi_i^*(\mathcal{S}_{it}) = \arg\max_{\pi_i \in \Pi} V^{\pi_i}(\mathcal{S}_{it}), \quad (5)$$

where Π is the set of stochastic policies.

4 Asynchronous supervised learning

4.1 Pre-training demonstration collection

The MDHDP $\pi'_i(\mathcal{S}_{it})$ drives vehicles to automatically collect real-world driving demonstrations, which is based on a simplified look-ahead scheme:

1. Steering angle s_{it}

Our data collection vehicle i determines the wheel steering angle s_{it} based on the current vehicle speed $\boldsymbol{v}_{it}$ and the position of the vehicle in front. Specifically, a target $F(x'_{it}, y'_{it})$ on the road's middle line is determined according to the following formula:

$$l_{EF} = L + v_{it} \cdot \triangle t, \quad (6)$$

where $E(x_{it}, y_{it})$ is the projection point of vehicle i's center on the road's middle line, l_{EF} is the length of road between points F and E, L is the constant look-ahead distance, and $\triangle t$ is the look-ahead factor. Then the wheel steering angle toward target F is computed as follows:

$$s_{it} = < \overrightarrow{JF}, \overrightarrow{v_{it}} >, \quad (7)$$

where $J(x^{\mathrm{c}}_{it}, y^{\mathrm{c}}_{it})$ is the position of vehicle i's center. To avoid collision with the vehicle in front, or vehicle j, s_{it} is modified according to the angle $< \overrightarrow{v_{jt}}, \overrightarrow{v_{it}} >$ between vehicles j and i:

$$s_{it} \leftarrow s_{it} \cdot W + 2(1 - W) \cdot < \overrightarrow{v_{jt}}, \overrightarrow{v_{it}} >, \quad (8)$$

where $W = D/C$ is the weight, D is the lateral space of vehicles j and i, and C is the side collision margin.

2. Break b_{it}

The brake b_{it} of vehicle i is determined according to its v_{it}, the speed limit $v^*_{r_t}$ of current road section r_t, and the distance d_{ijt} between vehicles j and i. The speed limit $v^*_{r_t}$ is determined by the curvature ρ of r_t, because different combinations of curvature and speed cause different centrifugal forces F_N, and F_N should be kept within a reasonable range for driving safety. $v^*_{r_t}$ is determined based on the relevant theory of aerodynamics:

$$v^*_{r_t} = \sqrt{\frac{g\mu(1/\rho)}{1 - \min\left(1, \frac{(1/\rho)\mathrm{CA}\mu}{M}\right)}}, \quad (9)$$

where g is the gravitational acceleration, μ is the friction coefficient, CA is the downforce coefficient, and M is the vehicle weight. If v_{it} exceeds the speed limit $v^*_{r_t}$ or vehicle i is too close to vehicle j, v_{it} needs to be decelerated, and the MDHDP just returns $b_{it} = 1$.

3. Throttle t_{it}

The throttle t_{it} of vehicle i is determined according to $v^*_{r_t}$ and v_{it}. If v_{it} is much less than $v^*_{r_t}$ and $b_{it} = 0$, MDHDP just returns $t_{it} = 1$.

In the TORCS (The Open Racing Car Simulator) simulation environment, data collection vehicles collect driving demonstrations online and perform the ASL pre-training process at the same time. Therefore, in the simulation verification process, the size of all thread-specific pre-training data sets is the same as the number of pre-training steps G_{pre}, so we collect G_{pre} demonstrations to pre-train our end-to-end autonomous driving model. However, in the engineering implementation process, due to the limitation of the small indoor driving demonstration scene, it is not possible to collect enough real-vehicle demonstration data. Therefore, we first collect a small batch of demonstration data, and then in the pre-training stage, we meet the requirement of pre-training steps by repeating a random sampling from the demonstration data set. In the real-vehicle verification process, these pre-training data sets are collected by the vehicle shown in Fig. 1.

4.2 Asynchronous supervised learning framework

We combine SL with the theory of RL to solve problem (5). We refer to an actor-critic network as a nonlinear function approximator to estimate the stochastic action policy $\mathcal{A}'_{in} \sim \pi_i(\mathcal{S}_{in}; \boldsymbol{\theta})$ and the value function $V^{\pi_i}(\mathcal{S}_{in}; \boldsymbol{\theta}_v)$, where $\boldsymbol{\theta}$ and $\boldsymbol{\theta}_v$ are the

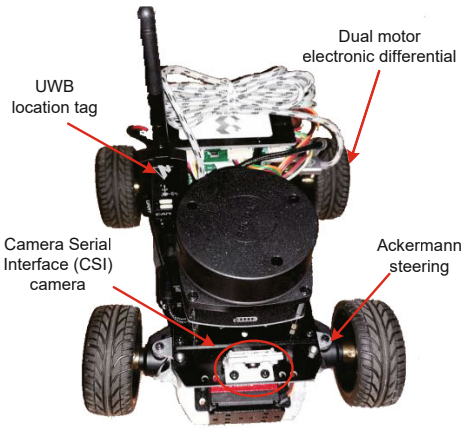


Fig. 1 The robot car used to collect demonstrations

parameters of actor and critic neural networks, respectively. We use Eq. (4) to update both $\boldsymbol{\theta}$ and $\boldsymbol{\theta}_v$. Our ASL framework relies on parallel actor-learners and accumulated updates for improving training stability. The update performed by multiple agents executing in parallel and asynchronously on multiple threads is as follows:

$$\begin{cases} \mathrm{d}\boldsymbol{\theta} \leftarrow \mathrm{d}\boldsymbol{\theta} + \bigtriangledown_{\boldsymbol{\theta}'} \log \pi_i(\mathcal{A}''_{in}|\mathcal{S}_{in};\boldsymbol{\theta}') A^{\pi_i}(\mathcal{S}_{in},\mathcal{A}''_{in}), \\ \mathrm{d}\boldsymbol{\theta}_v \leftarrow \mathrm{d}\boldsymbol{\theta}_v + \partial (A^{\pi_i}(\mathcal{S}_{in},\mathcal{A}''_{in}))^2/\partial \boldsymbol{\theta}'_v, \end{cases} \tag{10}$$

where $\boldsymbol{\theta}'$ and $\boldsymbol{\theta}'_v$ are thread-specific parameter vectors, and $\boldsymbol{\theta}$ and $\boldsymbol{\theta}_v$ are global shared parameter vectors. $\log \pi_i(\mathcal{A}''_{in}|\mathcal{S}_{in};\boldsymbol{\theta}')$ denotes the "surprise" that policy π_i selects action $\mathcal{A}''_{in}$. The smaller the probability that policy π_i selects $\mathcal{A}''_{in}$ is, the greater the "surprise" becomes. If the action with a low probability selected by policy π_i has a great advantage $A^{\pi_i}(\mathcal{S}_{in},\mathcal{A}''_{in})$, the actor network's parameters $\boldsymbol{\theta}$ will be more likely updated in this direction. The critic network updates its parameters $\boldsymbol{\theta}_v$ to minimize $A^{\pi_i}(\mathcal{S}_{in},\mathcal{A}''_{in})$ all the time, to improve the prediction of the state value function $V^{\pi_i}(\mathcal{S}_{in})$.

Fig. 2 shows the end-to-end autonomous driving model that we used, which takes four pre-processed images closest to the current moment as its inputs. Note that the parameters $\boldsymbol{\theta}$ of the stochastic action policy and $\boldsymbol{\theta}_v$ of the value function share some of the parameters in practice. This autonomous driving model uses a convolutional neural network that has two linear outputs for the mean and variance of the steering's normal probability distribution and one linear output for the value function $V^{\pi_i}(\mathcal{S}_{in};\boldsymbol{\theta}_v)$, with all non-output layers shared. Fig. 3 shows our methodological framework of ASL, and the pre-training process in real-world settings is given in Algorithm 1.

Our ASL-based pre-training method follows an offline multi-process implementation in real-world settings. That is, multiple SL training processes can be executed in a parallel and asynchronous offline manner, on multiple real-world driving demonstration data sets. After pre-training, we perform online RL training processes in real-world settings.

4.3 Complexity analysis

The pre-training algorithm includes significant image preprocessing, loss calculations, replay buffers, and actor-critic network approximators with six neural networks. Next, we analyze the time complexity (computations) and space complexity (memory) of the ASL-based pre-training process, wherein the time complexity is represented by the number of floating-point operations per second (FLOPS).

Algorithm 1 Pre-training with ASL in real-world settings

Initialize the number of pre-training steps G_{pre}, $\boldsymbol{\theta}$, and $\boldsymbol{\theta}_v$ with random weights, and the batching tuple size N
Initialize the thread-specific step counter $n \leftarrow 1$
Initialize the global step counter $G \leftarrow 1$
// Training
repeat
  Set $n \leftarrow 1$
  repeat
    Obtain tuple $(\mathcal{S}_{ik},\mathcal{A}_{ik})$ from the SL training data set Ω by random sampling
    Input state $\mathcal{S}_{ik}$, and perform action $\mathcal{A}''_{ik}$
    Calculate and store loss L_{ik} (Eq. (1))
    Store L_{ik} together with $\mathcal{S}_{ik}$ and $\mathcal{A}''_{ik}$ as the n^{th} tuple $(\mathcal{S}_{in},\mathcal{A}''_{in},L_{in})$ in the sampling batch
    $n \leftarrow n+1$
    $G \leftarrow G+1$
  until $n == N$
  for n=1 to N **do**
    Accumulate gradients $\boldsymbol{\theta}'$: $\mathrm{d}\boldsymbol{\theta} \leftarrow \mathrm{d}\boldsymbol{\theta} + \bigtriangledown_{\boldsymbol{\theta}'} \log \pi_i(\mathcal{A}''_{in}|\mathcal{S}_{in};\boldsymbol{\theta}') A^{\pi_i}(\mathcal{S}_{in},\mathcal{A}''_{in})$
    Accumulate gradients $\boldsymbol{\theta}'_v$: $\mathrm{d}\boldsymbol{\theta}_v \leftarrow \mathrm{d}\boldsymbol{\theta}_v + \partial (A^{\pi_i}(\mathcal{S}_{in},\mathcal{A}''_{in}))^2/\partial \boldsymbol{\theta}'_v$
  end for
  Perform asynchronous update of $\boldsymbol{\theta}$ using $\mathrm{d}\boldsymbol{\theta}$ and of $\boldsymbol{\theta}_v$ using $\mathrm{d}\boldsymbol{\theta}_v$, and then zero $\mathrm{d}\boldsymbol{\theta}$ and $\mathrm{d}\boldsymbol{\theta}_v$
  Synchronize thread-specific parameters $\boldsymbol{\theta}' = \boldsymbol{\theta}$ and $\boldsymbol{\theta}'_v = \boldsymbol{\theta}_v$
until $G > G_{\mathrm{pre}}$

Since the sampled driving demonstration tuple changes constantly, the image preprocessing has to be conducted at all steps during the pre-training stage. The time complexity of image preprocessing is $N(\mathcal{S}_{in}) \times m$, where $N(\mathcal{S}_{in})$ is the number of pixels of state $\mathcal{S}_{in}$ and m is the number of parallel actor-learners. To avoid repeated calculations, the algorithm has to record the means and standard deviations of the state variables, so the space complexity of image preprocessing is $2N(\mathcal{S}_{in}) \times m$. The experience replay buffer in ASL occupies some space to store $\mathcal{S}_{in}$; hence, the space complexity is all threads' batching tuple size $N \times m$.

For neural networks, the computation of the

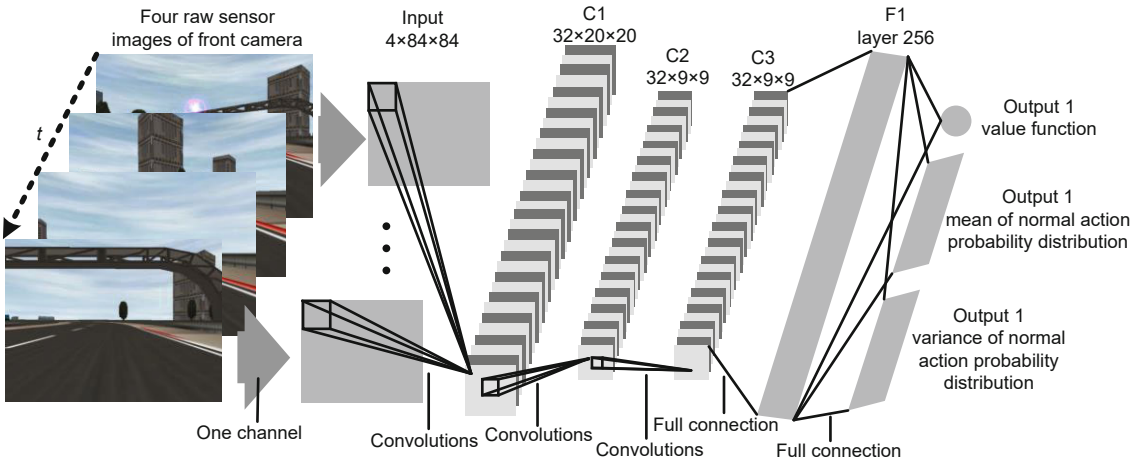


Fig. 2 Architecture of the end-to-end autonomous driving model

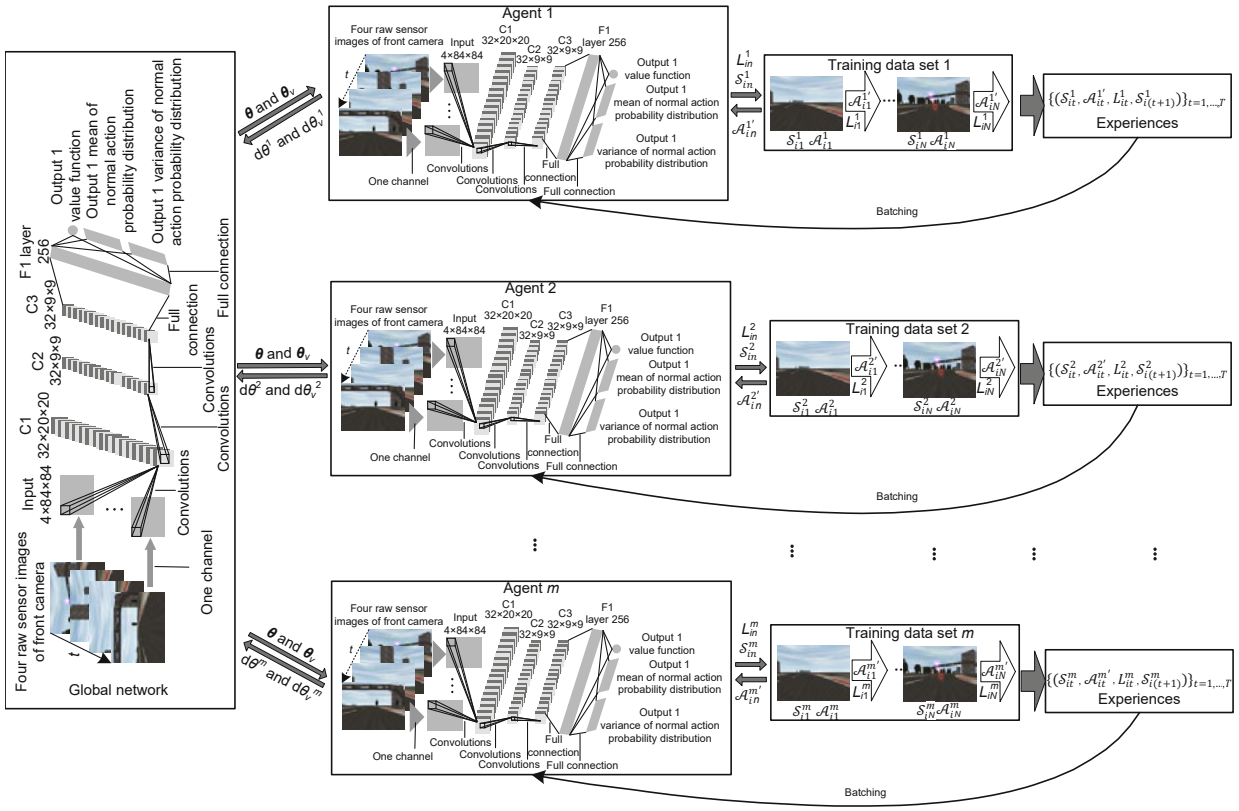


Fig. 3 Methodological framework of ASL: the presented pre-training method asynchronously executes multiple SL processes in parallel on multiple driving demonstration data sets

activation layers needs to be analyzed. When calculating FLOPS, addition, subtraction, multiplication, division, exponentiation, square root, and so on are usually counted as a single FLOPS. When there are X inputs for activation layers, the computational complexities of rectified linear unit (ReLU) layers, sigmoid layers, and tanh layers are X, $4X$, and $6X$, respectively (Qiu et al., 2019).

The input $\mathcal{S}_{in}$ is an image. Hence, there are convolutional layers in each actor-critic network. Assume that each actor-critic network has $C(Q)$ convolutional layers and $N(Q)$ fully connected layers. For convolutional layers, the total time complexity of all convolutional layers is (He KM and Sun, 2015)

$$O\left(\sum_{n=1}^{C(Q)} C_n \cdot C_{n-1} \cdot M_n^2 \cdot K_n^2\right), \tag{11}$$

where n is the index of a convolutional layer, C_n is the number of filters (also known as "width") in the n^{th} convolutional layer, C_{n-1} is also known as the number of input channels of the n^{th} layer, K_n is the spatial size (length) of the filter, and M_n is the spatial size of the output feature map.

For fully connected layers, considering the adding of bias, the time complexity can be calculated as follows:

$$\begin{aligned} &m\left(v_{\text{act}} \cdot \sum_{k=1}^{N(Q)-2} N(Q)_k + \sum_{k=0}^{N(Q)-1} N(Q)_k \cdot N(Q)_{k+1}\right) \\ &= O\left(\sum_{k=0}^{N(Q)-1} N(Q)_k \cdot N(Q)_{k+1}\right), \end{aligned} \tag{12}$$

where $N(Q)_k$ represents the unit number in the k^{th} fully connected layer, $N(Q)_0$ equals the output size of the $C(Q)^{\text{th}}$ convolutional layer, and v_{act} indicates the corresponding parameter determined by the type of the activation layer.

For a convolutional layer, there is a $C_n \times C_{n-1} \times K_n^2$ weight matrix and a C_n bias vector. In addition, there is a $C_n \times M_n^2$ output feature map. Hence, the memory of one convolutional layer is $C_n(M_n^2 + C_{n-1} \cdot K_n^2)$. Because the activation does not need to save weights, the space complexity of all convolutional layers is formulated as follows:

$$\begin{aligned} &m \sum_{n=1}^{C(Q)} C_n(M_n^2 + C_{n-1} \cdot K_n^2) \\ =&O\left(\sum_{n=1}^{C(Q)} C_n(M_n^2 + C_{n-1} \cdot K_n^2)\right). \end{aligned} \tag{13}$$

For a fully connected layer, there is an $N(Q)_k \times N(Q)_{k+1}$ weight matrix and an $N(Q)_{k+1}$ bias vector. Hence, the memory of one fully connected layer is $(N(Q)_k + 1)N(Q)_{k+1}$. The space complexity of all fully connected layers is formulated as follows:

$$\begin{aligned} &m \sum_{k=0}^{N(Q)-1} (N(Q)_k + 1)N(Q)_{k+1} \\ =&O\left(\sum_{k=0}^{N(Q)-1} N(Q)_k \cdot N(Q)_{k+1}\right). \end{aligned} \tag{14}$$

For each training step, the computation of SL loss is all threads' batching tuple size $N \times m$. Therefore, the overall time complexity of our training algorithm is represented as follows:

$$\begin{aligned} &O\left(\sum_{n=1}^{C(Q)} C_n \cdot C_{n-1} \cdot M_n^2 \cdot K_n^2\right) \\ &+ O\left(\sum_{k=0}^{N(Q)-1} N(Q)_k \cdot N(Q)_{k+1}\right) \\ &+ O(N(\mathcal{S}_{it})) + O(N), \end{aligned} \tag{15}$$

and the overall space complexity of our training algorithm is as follows:

$$\begin{aligned} &O\left(\sum_{n=1}^{C(Q)} C_n(M_n^2 + C_{n-1} \cdot K_n^2)\right) \\ &+ O\left(\sum_{k=0}^{N(Q)-1} N(Q)_k \cdot N(Q)_{k+1}\right) \\ &+ O(N(\mathcal{S}_{it})) + O(N). \end{aligned} \tag{16}$$

4.4 Visualization analysis

What knowledge does our end-to-end autonomous driving model learn from the MDHDP through pre-training? Which part of the input image plays a key role in the final output? What changes have taken place in the network's focus after pre-training? Visualization analysis for the specific influences of the input units in the neural network is

necessary to analyze the reason for performance improvement after pre-training. However, it is challenging to implement visual analysis for a neural network that outputs continuous results. In this subsection, motivated by Li LZ et al. (2018), we propose a visualization method based on the basic theory of single-variable analysis to analyze the changes of our neural network after pre-training. The results of visualization are presented in the form of heatmaps, which directly show the specific areas that have important impacts on results. By comparing the heatmaps before and after pre-training, we can intuitively find the improvements of the neural network brought about by pre-training.

Based on the basic theory of single-variable analysis, we change the gray value of pixel o while keeping other pixels unchanged. The change in pixel o is $\triangle o$, and the influence on layer φ's outputs in network $\boldsymbol{\theta}$ is as follows:

$$I_{\varphi}^{\boldsymbol{\theta}}(\triangle o) = w_{\varphi} I_{\varphi-1}(\triangle o) + b_{\varphi}, \tag{17}$$

where $\varphi \geq 2$, and if $\varphi = 1$, $I_{\varphi-1}(\triangle o) = \triangle o$. For a target network $\boldsymbol{\theta}$, its weights $\{w_1, w_2, \ldots, w_f\}$ and biases $\{b_1, b_2, \ldots, b_f\}$ are fixed (f is the number of neurons), so $I_{\varphi}^{\boldsymbol{\theta}}(\triangle o)$ depends only on $\triangle o$. In actual operations, the influence of each pixel is calculated by Algorithm 2.

Algorithm 2 Visualization analysis process

1: Initialize the pixel matrix of input image $\mathbf{IM}[L, W]$
2: Initialize influence matrix $\mathbf{IV} = \text{zeros}(L, W)$
3: **for** $o \in \mathbf{IM}$ **do**
4: $\quad I_{\varphi}^{\boldsymbol{\theta}}(\triangle o) = I_{\varphi}^{\boldsymbol{\theta}}(o + \triangle o) - I_{\varphi}^{\boldsymbol{\theta}}(o)$
5: $\quad \mathbf{IV} \leftarrow I_{\varphi}^{\boldsymbol{\theta}}(\triangle o)$
6: **end for**
7: $\mathbf{IV} \rightarrow$ heatmap

5 Simulation and experimental results

5.1 Simulation setting

In TORCS (Chen CY et al., 2015), our RL-based autonomous vehicle i is added to the simulation environment. This environment, which is based on an end-to-end autonomous driving model like Fig. 2, uses a convolutional layer with 32 filters of size 8×8 with stride 4, followed by a convolutional layer with 32 filters of size 4×4 with stride 2, followed by a convolutional layer with 32 filters of size 3×3 with stride 1, and followed by a fully connected layer with 256 hidden units. All four hidden layers are followed by ReLUs. This network has two sets of outputs: two linear outputs for the mean and variance of $\mathcal{A}'_{in}$'s normal probability distribution and one linear output for the value function $V^{\pi_i}(\mathcal{S}_{in}; \boldsymbol{\theta}_v)$. We compare our method with four other methods as follows:

1. Typical SL (TSL)

To demonstrate the superiority of our proposed ASL method to other SL methods, it is compared with an SL method that has been widely used in imitation learning tasks (Bojarski et al., 2016).

2. Asynchronous advantage actor-critic (A3C)

To demonstrate the effectiveness of our pre-training method, it is compared with a recently developed deep RL algorithm without pre-training, the A3C (Mnih et al., 2016).

3. A3C with pre-trained convolutional layers (A3C-PCL)

To demonstrate the superiority of our pre-training method ASL to other pre-training algorithms, we compare it with a recently developed pre-training algorithm, i.e., the convolutional layer pre-training method (de la Cruz et al., 2019).

4. Look-ahead scheme (LAS)

To prove that our RL-based model has better performance than the existing traditional autonomous driving schemes, the LAS is compared with our end-to-end autonomous driving model.

In the RL training stage, we consider vehicle damage, stuck, tangential velocity along the lane, and distance from the lane centerline, to formulate the RL reward by the following expression:

$$R_{it} = \begin{cases} c, & \text{get damaged or stuck,} \\ z, & e \leq z_1 \text{ or } d \geq z_2, \\ \lambda e - \beta d, & \text{otherwise,} \end{cases} \tag{18}$$

where λ and β are two positive weights, e denotes the tangential velocity in the t^{th} time slot, d denotes the distance from lane centerline, and c and z are two negative constants. We report the initial performance, convergence step, and the best performance of the RL-based agent and also measure the performance improvement using two metrics adapted from Taylor and Stone (2009) and de la Cruz et al. (2019):

(1) Initial performance: the average reward obtained for the first time by the agent in the RL training stage.

(2) Convergence step: when the average reward's standard deviation obtained by the agent in the RL training stage for 5×10^6 consecutive training steps does not exceed 500, it can be considered that this model has converged, and the initial step of this convergence interval is the convergence step.

(3) Best performance: the highest average reward obtained by the agent in the RL training stage.

(4) Final performance: the final learned performance of the agent in the RL training stage. We use the reward obtained at step 2×10^7 as the value for the final performance.

(5) Total performance: the total reward accumulated (i.e., the area under the learning curve (AUC)) by the agent. We approximate the AUC using the trapezoidal rule: AUC $\approx \sum_{s=1}^{S}\frac{f(x_{s-1})+f(x_s)}{2}\triangle x_s$, where $f(x_s)$ is the reward value at step s and $\triangle x_s = x_s - x_{s-1} = 1\times10^3$ is the evaluation frequency.

The parameter settings are given in Table 1.

5.2 Performance in the pre-training stage

In this subsection, the focus is on the performance of our proposed ASL method in the pre-training stage. First, we analyze the change of the relationship between the output result and model input, to demonstrate the effectiveness of pre-training from the detailed point of view. The visualization results before and after pre-training are shown in Fig. 4, in which white indicates pixels that are important for current decision making, while black indicates unimportant pixels. The top row shows the raw inputs; the second row shows the visualization results of our RL-based agent before pre-training; the bottom row shows the visualization results of the agent after pre-training. It can be seen that the focus of the agent before pre-training is meaningless from the second row of Fig. 4. In particular, when the agent is initialized, it is interested in all elements of the raw input (top row of Fig. 4). After pre-training, the agent's focus is meaningful from the bottom row of Fig. 4. It can be seen that the pre-trained agent pays more attention to the factors that affect driving decision making such as the road (red-orange) than to some irrelevant factors (black). The proposed visualization method helps us better understand the internal mechanism of deep RL decision making from the detailed point of view, and demonstrates the effectiveness of pre-training in improving the performance of a deep RL-based model.

We then analyze the convergence performance of ASL and another SL method in the pre-training stage. Fig. 5 shows the loss curves of our model trained by ASL and TSL, respectively. For the ASL method, it can be seen that the loss can converge

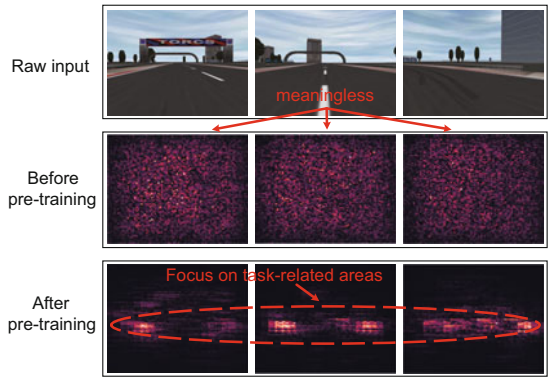


Fig. 4 Visualization results: the top row shows the raw inputs of our RL-based model; the second and bottom rows show the visualization results of our agent before and after pre-training, respectively (References to color refer to the online version of this figure)

Table 1 Simulation parameters

Parameter	Value
Discount factor γ	0.99
Learning rate α	0.000 45
Batching tuple size N	10
Number of pre-training steps G_{pre}	2×10^7
Loss parameters ς, ϱ ($\|\mathcal{A}_{in}-\mathcal{A}'_{in}\|_1>1.5$)	0, 0.02
Loss parameters ς, ϱ ($1.5\geq\|\mathcal{A}_{in}-\mathcal{A}'_{in}\|_1>1.0$)	0, 0.01
Loss parameters ς, ϱ ($1.0\geq\|\mathcal{A}_{in}-\mathcal{A}'_{in}\|_1>0.5$)	0.02, 0.02
Loss parameters ς, ϱ ($0.5\geq\|\mathcal{A}_{in}-\mathcal{A}'_{in}\|_1$)	0.0225, 0.015
Reward parameters c, z, z_1, z_2, λ, β	-0.03, -0.02, 2.5, 1.8, 0.006, 0.0018
Number of parallel actor-learners m	4

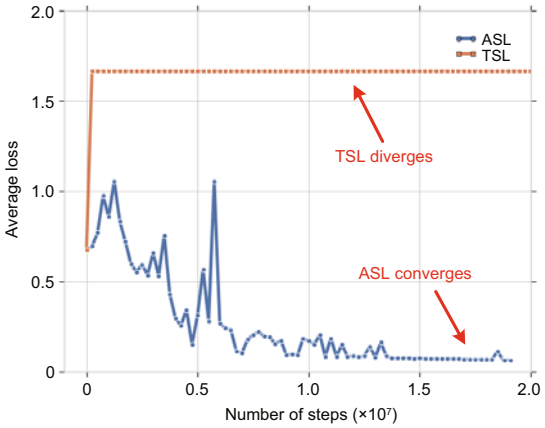


Fig. 5 Convergence performance in the pre-training stage, showing the superiority of our proposed ASL method compared with another SL method in convergence

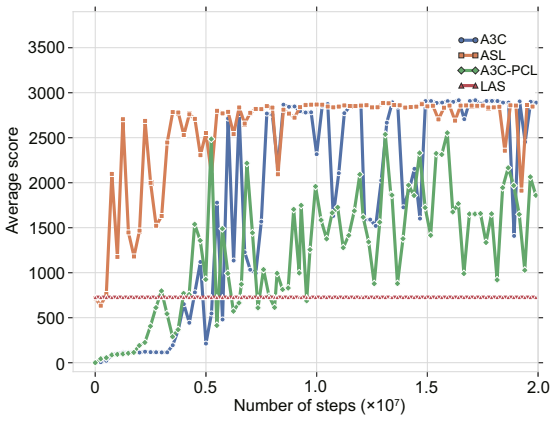


Fig. 6 Simulation results for different methods in the RL training stage, showing the reward against the total RL training step (References to color refer to the online version of this figure)

to a steady state (about 0.1) after about 1×10^7 steps, which proves that our proposed ASL method can make the RL-based model effectively learn the prior knowledge from the training data set. For the TSL method, it remains in a high loss state. TSL cannot adapt to the high-dimensional input or continuous output of the autonomous driving scenario, and therefore loses the ability to converge. This verifies the superiority of our proposed ASL method in imitation learning of autonomous driving model's pre-training tasks.

5.3 Performance in the RL training stage

In this subsection, the focus is on the performance of the pre-trained autonomous driving model in the RL training stage. We analyze the effectiveness of our pre-training method in improving subsequent RL training in real-world settings (simulated by TORCS). From Fig. 6, it can be seen that the autonomous driving model pre-trained by our ASL method with 1.0×10^7 pre-training steps converges faster in the RL training process than the A3C and A3C-PCL methods. More importantly, it does not have to suffer from poor initial performance. The red horizontal line shows the average LAS reward, which proves that our RL-based model performs better than the typical autonomous driving method. Interestingly, the performance of A3C-PCL is even worse than that of the A3C without pre-training. The A3C-PCL method divides the autonomous driving model into two parts, namely, the feature extractor and the controller, and pre-trains only the convolutional layers as a feature extractor. As described in the previous section, it is not possible to make a clean break between the part of the end-to-end network that functions primarily as a feature extractor and the part that serves as a controller. The simulation results show that training the convolutional layers alone, instead of training the entire network, destroys the integrity of the autonomous driving model, and causes its performance in the RL training stage to be inferior to that of the model without pre-training.

Table 2 shows the quantitative performance improvements of our pre-training ASL method over the baseline A3C and another pre-training method, A3C-PCL. The ASL method achieves the best performance in the RL training process with a remarkable total performance improvement of 36.07% over the A3C method. The convergence step of our proposed method is reduced by 51.43% compared to that of the A3C method. Compared with the A3C method, the total performance of the A3C-PCL method decreases by 39.34%. In addition, the A3C-PCL method has different effects on neural networks of different structures, so it is difficult to use it as a general methodological framework for autonomous driving model's pre-training tasks. The quantitative results verify the effectiveness of our proposed ASL method in improving the autonomous driving model performance in the RL training stage, and prove the superiority of our proposed joint methodological framework of RL and SL.

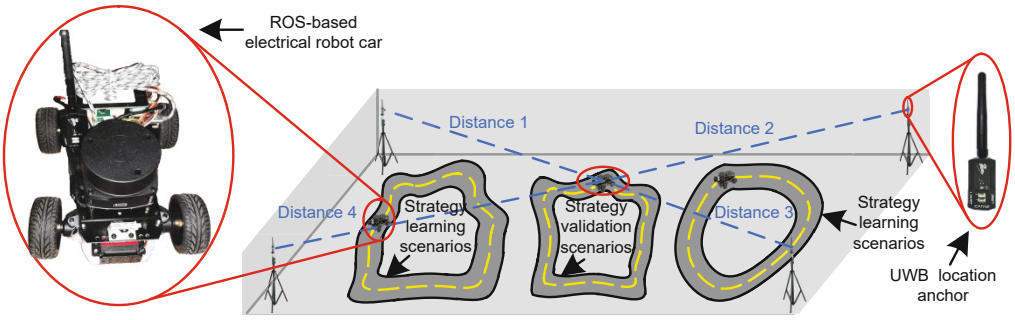
5.4 Real test

To prove the feasibility of our proposed pre-training method in real-world settings, we built a real-vehicle verification system (shown in Fig. 7) and tested it with three electrical robot cars (shown in Fig. 1) equipped with a Camera Serial Interface (CSI) camera. Note that to obtain accurate RL reward values (Eq. (18)), we used an ultra wide band (UWB) system to measure the state values of each vehicle. Fig. 8 shows the loss curve of the ASL method in the pre-training stage. It can be seen that the loss can converge to a steady state (about 0.05) after about 4×10^7 steps. After pre-training, the RL-based model learned prior knowledge of driving in real-world settings before RL training, which would improve the convergence speed of RL training. Then we can import the pre-trained model parameters into the end-to-end autonomous driving models of these vehicles in the real-vehicle verification system. The training system would asynchronously execute multiple RL processes with three vehicles in parallel, on multiple tracks (different driving environments). Like our proposed ASL method, this asynchronous

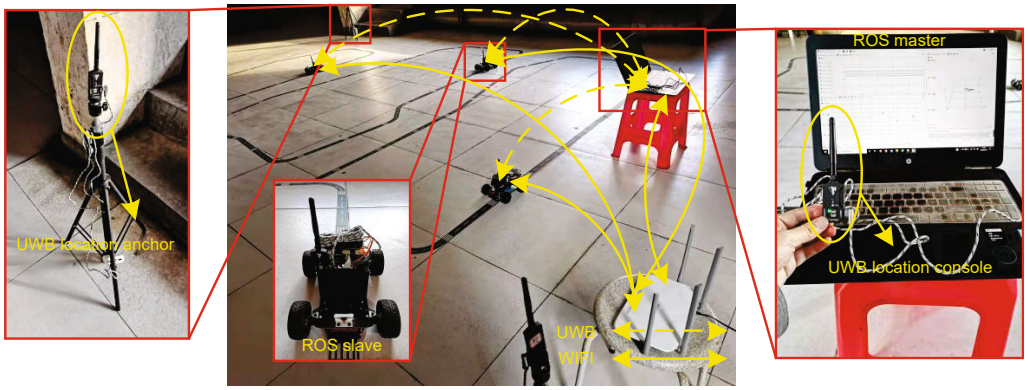
Table 2 Quantitative evaluations of different methods using five metrics: initial performance, convergence step, best performance, final performance, and total performance

Performance metric	ASL	A3C	A3C-PCL	LAS
Initial performance	715.83 ± 0.00	-0.13 ± 0.05	-0.15 ± 0.02	728.16
Convergence step ($\times10^7$)	0.51 ± 0.30	1.05 ± 0.99	0.18 ± 0.32	–
Best performance ($\times10^3$)	2.89 ± 0.01	2.92 ± 0.00	2.38 ± 0.20	0.73
Final performance ($\times10^3$)	1.98 ± 0.74	1.47 ± 1.44	0.14 ± 0.24	0.73
Total performance ($\times10^{10}$)	4.98 ± 0.14	3.66 ± 0.11	2.22 ± 0.21	0.29

Results for ASL, A3C, and A3C-PCL are represented as mean±standard deviation



(a)



(b)

Fig. 7 The real-vehicle verification system: (a) the schematic diagram of real-vehicle scenarios; (b) the physical picture of real-vehicle scenarios

training method can stabilize the training process and accelerate convergence.

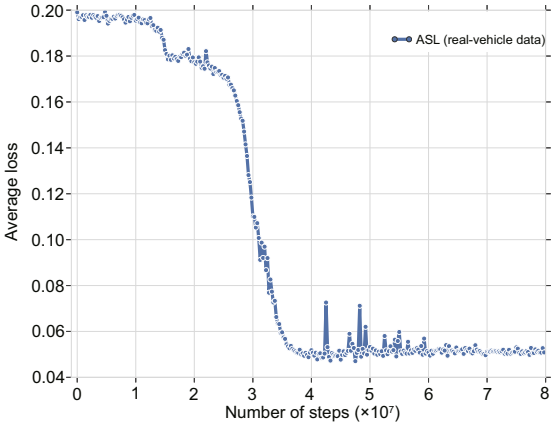


Fig. 8 Test result with real-world driving demonstration data

6 Conclusions

Although many advanced RL algorithms have been developed for autonomous driving, such as A3C and DDPG, they all suffer from poor initial performance. We have proposed an ASL method to improve the convergence speed of the autonomous driving model in an RL training process, and our trained model can achieve better performance than the typical autonomous driving methods. We have found a feasible and effective visualization method to analyze the improvement due to pre-training from the detailed point of view, which uses heatmaps to visualize the specific influences of the input units in the neural network. The visualization results are meaningful and explicable, which helps us determine if the pre-trained model has learned prior knowledge. Simulation and experimental results have demonstrated the feasibility, effectiveness, and superiority of our method from both the microscopic and macroscopic points of view.

We will explore more different RL theories to improve the performance of our proposed joint RL and SL methodological framework. The next big step is to train the end-to-end autonomous driving model in real urban traffic situations, such as intersections and T junctions. Additional future work is, if legal issues are resolved, to verify our RL-based system on public highways without driver intervention.

Contributors

Yunpeng WANG designed the research. Kunxian ZHENG processed the data. Daxin TIAN drafted the manuscript. Xuting DUAN helped organize the manuscript. Kunxian ZHENG and Jianshan ZHOU revised and finalized the paper.

Compliance with ethics guidelines

Yunpeng WANG, Kunxian ZHENG, Daxin TIAN, Xuting DUAN, and Jianshan ZHOU declare that they have no conflict of interest.

References

Bai ZW, Shangguan W, Cai BG, et al., 2019. Deep reinforcement learning based high-level driving behavior decision-making model in heterogeneous traffic. Proc Chinese Control Conf, p.8600-8605. https://doi.org/10.23919/ChiCC.2019.8866005

Bojarski M, Del Testa D, Dworakowski D, et al., 2016. End to end learning for self-driving cars. https://arxiv.org/abs/1604.07316

Brys T, Harutyunyan A, Suay HB, et al., 2015. Reinforcement learning from demonstration through shaping. Proc 24th Int Conf on Artificial Intelligence, p.3352-3358.

Chen CY, Seff A, Kornhauser A, et al., 2015. DeepDriving: learning affordance for direct perception in autonomous driving. Proc IEEE Int Conf on Computer Vision, p.2722-2730. https://doi.org/10.1109/ICCV.2015.312

Chen JY, Yuan BD, Tomizuka M, 2019. Model-free deep reinforcement learning for urban autonomous driving. Proc IEEE Intelligent Transportation Systems Conf, p.2765-2771. https://doi.org/10.1109/ITSC.2019.8917306

Codevilla F, Müller M, López A, et al., 2018. End-to-end driving via conditional imitation learning. Proc IEEE Int Conf on Robotics and Automation, p.4693-4700. https://doi.org/10.1109/ICRA.2018.8460487

de la Cruz GVJr, Du YS, Taylor ME, 2019. Pre-training with non-expert human demonstration for deep reinforcement learning. *Knowl Eng Rev*, 34:e10. https://doi.org/10.1017/S0269888919000055

González D, Pérez J, Milanés V, et al., 2016. A review of motion planning techniques for automated vehicles. *IEEE Trans Intell Transp Syst*, 17(4):1135-1145. https://doi.org/10.1109/TITS.2015.2498841

Hao W, Lin YJ, Cheng Y, et al., 2018. Signal progression model for long arterial: intersection grouping and coordination. *IEEE Access*, 6:30128-30136. https://doi.org/10.1109/ACCESS.2018.2843324

He KM, Sun J, 2015. Convolutional neural networks at constrained time cost. Proc IEEE Conf on Computer Vision and Pattern Recognition, p.5353-5360. https://doi.org/10.1109/CVPR.2015.7299173

He Y, Zhao N, Yin HX, 2018. Integrated networking, caching, and computing for connected vehicles: a deep reinforcement learning approach. *IEEE Trans Veh Technol*, 67(1):44-55. https://doi.org/10.1109/TVT.2017.2760281

Li L, Lv YS, Wang FY, 2016. Traffic signal timing via deep reinforcement learning. *IEEE/CAA J Autom Sin*, 3(3):247-254. https://doi.org/10.1109/JAS.2016.7508798

Li LZ, Ota K, Dong MX, 2018. Humanlike driving: empirical decision-making system for autonomous vehicles. *IEEE Trans Veh Technol*, 67(8):6814-6823. https://doi.org/10.1109/TVT.2018.2822762

Liu N, Li Z, Xu JL, et al., 2017. A hierarchical framework of cloud resource allocation and power management using deep reinforcement learning. Proc IEEE 37th Int Conf on Distributed Computing Systems, p.372-382. https://doi.org/10.1109/ICDCS.2017.123

Mao HZ, Alizadeh M, Menache I, et al., 2016. Resource management with deep reinforcement learning. Proc 15th ACM Workshop on Hot Topics in Networks, p.50-56. https://doi.org/10.1145/3005745.3005750

Mnih V, Kavukcuoglu K, Silver D, et al., 2013. Playing Atari with deep reinforcement learning. https://arxiv.org/abs/1312.5602

Mnih V, Kavukcuoglu K, Silver D, et al., 2015. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529-533. https://doi.org/10.1038/nature14236

Mnih V, Badia AP, Mirza M, et al., 2016. Asynchronous methods for deep reinforcement learning. Proc 33rd Int Conf on Machine Learning, p.1928-1937.

Nair A, Srinivasan P, Blackwell S, et al., 2015. Massively parallel methods for deep reinforcement learning. https://arxiv.org/abs/1507.04296

Nair A, McGrew B, Andrychowicz M, et al., 2018. Overcoming exploration in reinforcement learning with demonstrations. https://arxiv.org/abs/1709.10089

Paden B, Čáp M, Yong SZ, et al., 2016. A survey of motion planning and control techniques for self-driving urban vehicles. *IEEE Trans Intell Veh*, 1(1):33-55. https://doi.org/10.1109/TIV.2016.2578706

Qiu CR, Hu Y, Chen Y, et al., 2019. Deep deterministic policy gradient (DDPG)-based energy harvesting wireless communications. *IEEE Int Things J*, 6(5):8577-8588. https://doi.org/10.1109/JIOT.2019.2921159

Sallab AE, Abdou M, Perot E, et al., 2017. Deep reinforcement learning framework for autonomous driving. *Electron Imag*, 2017(19):70-76. https://doi.org/10.2352/ISSN.2470-1173.2017.19.AVM-023

Schwarting W, Alonso-Mora J, Rus D, 2018. Planning and decision-making for autonomous vehicles. *Ann Rev Contr Robot Auton Syst*, 1:187-210. https://doi.org/10.1146/annurev-control-060117-105157

Selvaraju RR, Cogswell M, Das A, et al., 2019. Grad-CAM: visual explanations from deep networks via gradient-based localization. *Int J Comput Vis*, 128(8):336-359. https://doi.org/10.1007/s11263-019-01228-7

Silver D, Schrittwieser J, Simonyan K, et al., 2017. Mastering the game of Go without human knowledge. *Nature*, 550(7676):354-359. https://doi.org/10.1038/nature24270

Taylor ME, Stone P, 2009. Transfer learning for reinforcement learning domains: a survey. *J Mach Learn Res*, 10:1633-1685.

Wang YP, Zheng KX, Tian DX, et al., 2020. Cooperative channel assignment for VANETs based on multiagent reinforcement learning. *Front Inform Technol Electron Eng*, 21(7):1047-1058. https://doi.org/10.1631/FITEE.1900308

Xu ZY, Wang YZ, Tang J, et al., 2017. A deep reinforcement learning based framework for power-efficient resource allocation in cloud RANs. Proc IEEE Int Conf on Communications, p.1-6. https://doi.org/10.1109/ICC.2017.7997286

Zhang XQ, Ma HM, 2018. Pretraining deep actor-critic reinforcement learning algorithms with expert demonstrations. https://arxiv.org/abs/1801.10459

Zhou BL, Khosla A, Lapedriza A, et al., 2016. Learning deep features for discriminative localization. Proc IEEE Conf on Computer Vision and Pattern Recognition, p.2921-2929. https://doi.org/10.1109/CVPR.2016.319