2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 25-29, 2020, Las Vegas, NV, USA (Virtual)

Hierarchical Reinforcement Learning Method for Autonomous Vehicle Behavior Planning*

Zhiqian Qiao[1], Zachariah Tyree[2], Priyantha Mudalige[2], Jeff Schneider[3] and John M. Dolan[3]

***Abstract*— Behavioral decision making is an important aspect of autonomous vehicles (AV). In this work, we propose a behavior planning structure based on hierarchical reinforcement learning (HRL) which is capable of performing autonomous vehicle planning tasks in simulated environments with multiple sub-goals. In this hierarchical structure, the network is capable of 1) learning one task with multiple sub-goals simultaneously; 2) extracting attentions of states according to changing sub-goals during the learning process; 3) reusing the well-trained network of sub-goals for other tasks with the same sub-goals. A hybrid reward mechanism is designed for different hierarchical layers in the proposed HRL structure. Compared to traditional RL methods, our algorithm is more sample-efficient, since its modular design allows reusing the policies of sub-goals across similar tasks for various transportation scenarios. The results show that the proposed method converges to an optimal policy faster than traditional RL methods.**

I. INTRODUCTION

In a traditional AV system, after receiving the processed observations coming from the perception system, the ego vehicle performs behavior planning to deal with different scenarios. At the behavior planning level, algorithms generate high-level decisions such as *Go*, *Stop*, etc., and a lower-level trajectory planning system maps these high-level decisions to trajectories. Then a lower-level controller outputs the detailed pedal or brake inputs to allow the vehicle to follow these trajectories.

At first glance, among algorithms generating behavior decisions, heuristic-based rule-enumeration [1][2] appears to describe human-like decision processes well. However, adjusting the corresponding decisions to account for changes in the environment is difficult if the decisions of the AV are completely handcrafted. The rules need to be added one by one in order to respond to various situations. The environment can vary across different dimensions, all relevant to the task of driving, and the number of rules necessary for planning in this nuanced setting can be unmanageable. For example, when a vehicle is approaching a stop-line intersection with moving front vehicles, it should pay attention to both the front vehicle and stop-line. Rules prescribing how to stop or how to follow the front vehicle can be designed with tuning parameters of the distance to follow, time to decelerate, etc., and meanwhile the rules for prioritizing following front vehicle or decelerating to stop also need extra parameter-tuning work. A simple driving scenario for human drivers can thus result in time-consuming design in order to make the autonomous vehicle reach pre-defined goals.

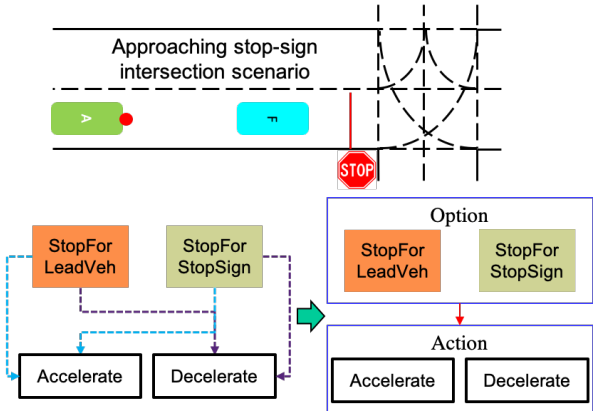

Fig. 1: Heuristic-based structure vs. HRL-based structure

An alternative method is reinforcement learning [3][4][5]. Suitable applications within the autonomous vehicle domain include learning an output controller for lane-following, merging into a roundabout, traversing an intersection and lane changing. However, low stability and large computational requirements make RL difficult to use widely for more general tasks with multiple sub-goals. Obviously, applying RL to learn the behavior planning system from scratch not only increases the difficulties of adding or deleting sub-functions within the existing behavior planning system, but also makes it harder to validate safety.

In traditional RL approaches it is necessary to train unique policies for different tasks. In order to train a new task the entire policy must be relearned regardless of how similar the two tasks may be. As a result, a hierarchical structure which is structurally similar to the heuristic-based algorithms is more feasible and can save computation time by learning different functions or tasks separately. Our goal in this work is to construct a single planning algorithm based on hierarchical reinforcement learning (HRL) which can accomplish behavior planning in an environment where the agent pursues multiple sub-goals and to do so in such a way that sub-goal policies can be reused for subsequent tasks in a modular fashion (see Fig. 1). The main contributions of the work are:

- A HRL structure with embedded state attention model.
- A hybrid reward function mechanism which can efficiently evaluate the performance for state-action pairs of different hierarchical levels.
- An efficient RL exploration method with simple-heuristic initialization.
- A hierarchical prioritized experience replay process.

[1]Zhiqian Qiao is a Ph.D. student of Electrical and Computer Engineering, Carnegie Mellon University, 5000 Forbes Ave, Pittsburgh, USA zhiqianq@andrew.cmu.edu
[2] Research & Development, General Motors
[3] Faculties of The Robotics Institute, Carnegie Mellon University

II. Related Work

This section summarizes previous work related to this paper, which can be categorized as: 1) papers that propose self-driving behavior planning algorithms; 2) papers that address reinforcement learning (RL) and hierarchical reinforcement learning (HRL) algorithms

A. Behavior Planning of Autonomous Vehicles

[6] proposed a slot-based approach to check if a situation is safe to merge into lanes or across an intersection with moving traffic, which is an example of a heuristic-based rule-enumeration method. This kind of algorithm needs significant effort by human designers to create rules that can deal with different scenarios, especially in urban environments. As a result, learning-based algorithms, especially reinforcement learning [7][8], have been applied to design policies. [9] formulated the decision-making problem for AV under uncertain environments as a POMDP and trained out a Bayesian Network to deal with a T-shape intersection merging problem. [10] dealt with the traversing intersection problem via Deep Q-Networks combined with a long-term memory component. [11] used Deep Recurrent Q-network (DRQN) with states from a bird's-eye view of the intersection to learn a policy for traversing the intersection. [12] proposed an efficient strategy to navigate through intersections with occlusion by using the DRL method. These works focused on designing variants of the state-space and add-on network modules in order to allow the agent to handle different scenarios.

B. Reinforcement Learning

Algorithms with extended functions based on RL and HRL have been proposed. [13] proposed the idea of a meta controller, which is used to define a policy governing when the lower-level action policy is initialized and terminated. [14] introduced the concept of hierarchical Q learning called MAXQ, which proved the convergence of MAXQ mathematically and could be computed faster than the original Q learning experimentally. [15] proposed an improved MAXQ method by combining the R-MAX [16] algorithm with MAXQ. It has both the efficient model-based exploration of R-MAX and the opportunities for abstraction provided by the MAXQ framework. [17] used the idea of the hierarchical model and transferred it into parameterized action representations. They use a DRL algorithm to train high-level parameterized actions and low-level actions together in order to get more stable results than by getting the continuous actions directly.

In our work, the main idea is to combine the heuristic-based decision-making structure with the HRL-based approaches in order to combine the advantages of both methods. We build the HRL-structure according to the heuristic method (see Fig. 1) so that the system can more easily figure out the local optimal policy based on local option choice and environment state. Meanwhile, it can allow the validation of different local policies within the hierarchical system instead of presenting a monolithic neural-network black-box policy.

III. Preliminaries

In this section, the preliminary background of the problem is described. The fundamental algorithms including Deep Q-Learing (DQN), Double Deep Q-Learning (DDQN) and Hierarchical Reinforcement Learning (HRL) are introduced.

1) Deep Q-learning and Double Deep Q-learning: Since being proposed, DQN [3] and DDQN [18] have been widely applied in reinforcement learning problems. In Q-learning, an action-value function $Q_\pi(s,a)$ is learned to get the optimal policy π which can maximize the action-value function $Q^*(s,a)$. s and a are current state and action, respectively. Hence, a parameterized action-value function $Q(s,a|\theta)$ is used with a discount factor γ, as in Equation 1 where r is the reward achieved when performing a based on s.

$$\begin{aligned} Q^*(s,a) &= \max_\theta Q(s,a|\theta) \\ &= r + \gamma \max_\theta Q(s',a'|\theta) \end{aligned} \tag{1}$$

2) Double Deep Q-learning: For the setting of DQN, the network parameter θ is optimized by minimizing the loss function $L(\theta)$, which is defined as the difference between the predicted action-value Q and the target action-value Y^Q. θ can be updated with a learning rate α, as shown in Equation 2. R_t means the reward received at time t.

$$\begin{aligned} Y_t^Q &= R_{t+1} + \gamma \max_a Q(S_{t+1}, a|\theta_\mathbf{t}) \\ L(\theta) &= \left(Y_t^Q - Q(S_t, A_t|\theta_t)\right)^2 \\ \theta_{t+1} &= \theta_t + \alpha \frac{\partial L(\theta)}{\partial \theta} \end{aligned} \tag{2}$$

For the DDQN, the target action-value Y^Q is revised according to another target Q-network $Q^{'}$ with parameter θ':

$$Y_t^Q = R_{t+1} + \gamma Q(S_{t+1}, \arg\max_a Q'(S_{t+1}, a|\theta_t)|\theta_t') \tag{3}$$

During the training procedure, techniques such as ε-greedy [19] and prioritized experience replay [20] can be applied to improve the training performance.

3) Hierarchical Reinforcement Learning: For the HRL model [13] with sequential sub-goals, a meta controller Q^1 generates the sub-goal g for the following steps and a controller Q^2outputs the actions based on this sub-goal until the next sub-goal is generated by the meta controller. N is the number of steps between the last time this controller was activated and the current one.

$$\begin{aligned} Y_t^{Q^1} &= \sum_{t'=t+1}^{t+1+N} R_{t'} + \gamma \max_g Q(S_{t+1+N}, g|\theta_\mathbf{t}{}^1) \\ Y_t^{Q^2} &= R_{t+1} + \gamma \max_a Q(S_{t+1}, a|\theta_\mathbf{t}^\mathbf{2}, g) \end{aligned} \tag{4}$$

IV. Methodology

In this section we present our proposed model, which is a hierarchical RL network with an explicit attention model, hybrid reward mechanism and a hierarchical prioritized experience replay training schema. We will refer to this model as Hybrid HRL throughout the paper.

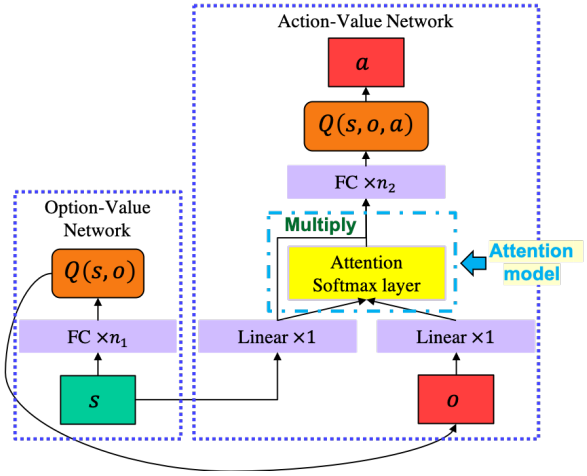

Fig. 2: Hierarchical RL Option and Action Q-Network. FC stands for a fully connected layer.

A. Hierarchical RL with Attention

Hierarchical structures based on RL can be applied to learn a task with multiple sub-goals. For a hierarchical structure with two levels, an option set O is assigned to the first level, whose object is to select among sub-goals. The weight θ_t^o is updated according to Equation 5.

$$\begin{aligned} O_{t+1}^* &= \arg\max_o Q^o(S_{t+1}, o|\theta_t^o) \\ Y_t^{Q^o} &= R_{t+1}^o + \gamma Q^o(S_{t+1}, O_{t+1}^*|\theta_t^{o'}) \\ L(\theta^o) &= \left(Y_t^{Q^o} - Q^o(S_t, O_t|\theta_t^o)\right)^2 \end{aligned} \tag{5}$$

After selecting an option o, the corresponding action set A^o represents the action candidates that can be executed on the second level of the hierarchical structure with respect to the selected option o. In many situations, the portion of the state set and the amount of abstraction needed to choose actions at different levels of this hierarchy can vary widely. In order to avoid designing a myriad of state representations corresponding to each hierarchy level and sub-goal, we share one state set S for the whole hierarchical structure. Meanwhile, an attention model is applied to define the importance of each state element $I(s,o)$ with respect to each hierarchical level and sub-goal and then use these weights to reconstruct the state s^I. The weight θ_t^a is updated accordingly to minimize $L(\theta^a)$ in Equation 6.

$$\begin{aligned} A_{t+1}^* &= \arg\max_a Q^a(S_{t+1}^{I(s,o)}, O_{t+1}^*, a|\theta_t^a) \\ Y_t^{Q^a} &= R_{t+1}^a + \gamma Q^a(S_{t+1}^{I(s,o)}, O_{t+1}^*, A_{t+1}^*|\theta_t^{o'}) \\ L(\theta^a) &= \left(Y_t^{Q^a} - Q^a(S_t^{I(s,o)}, O_t, A_t|\theta_t^a)\right)^2 \end{aligned} \tag{6}$$

When implementing the attention-based HRL, we construct the option network and the action network (Fig. 2), which includes the attention mechanism as a *softmax* layer in the action-value network Q^a.

B. Hybrid Reward Mechanism

Instead of generating one reward function which is applied to evaluate the final outputs coming from both options and actions in one step together, we designed a reward mechanism which can evaluate the goodness of option and action separately, as well as the goodness of the interaction between option and action. A hybrid reward mechanism is introduced: 1) according to the chosen option or action, the algorithm decides which reward function should be triggered; 2) meanwhile, a positive reward which benefits both option and action occurs if and only if the whole task and the sub-goals in the hierarchical structure have all been completed. Fig. 3 demonstrates the hybrid reward mechanism.

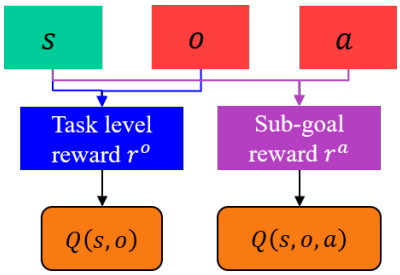

Fig. 3: Hybrid Reward Mechanism

C. Exploration with simple heuristic

Some recent work [21][22] proposed to use demonstration knowledge in the RL so that the algorithms can achieve more efficient learning. For the AV decision-making problem, heuristic-based methods can provide imperfect but successful solutions to finish assigned tasks with less exploration. As a result, inspired by the ε-greedy approach, we introduce the ε-HeuristicExplore (Eq. 7) approach, which explores the heuristic-based policies with higher probability during the early training stage and meanwhile performs random exploration with lower probability.

$$a = \begin{cases} \text{heuristic-based action} & \text{with probability } \varepsilon/2 \\ \text{random action} & \text{with probability } \varepsilon/2 \\ a* & \text{with probability } 1-\varepsilon \end{cases} \tag{7}$$

$a*$ is the action received from exploitation. With this approach, the agent can get a higher average reward at the beginning of training and less effective policies are stored in the training replay buffer. The agent can access both high-quality (heuristic-based) and random explorations during the whole training procedure. Instead of exploring an unknown environment with totally random actions, the agent gets an idea of what action may bring higher reward based on a rule-based algorithm, which helps the agent to learn more quickly.

D. Hierarchical Prioritized Experience Replay

[20] proposed a framework to efficiently replay experience for DQN so that the stored transitions $\{s,a,r,s'\}$ with higher temporal difference error (TD-error) in the previous training iteration result in a higher probability of being selected in the mini-batch for the current iteration. However, in the HRL structure, the rewards received from the whole system not only rely on the current level, but also are affected by the interactions among different hierarchical

Algorithm 1 Hierarchical RL with Attention State

procedure HRL-AHR()
  Initialize option and action network Q^o, Q^a with weights θ^o, θ^a and the target option and action network $Q^{o'}$, $Q^{a'}$ with weights $\theta^{o'}$, $\theta^{a'}$.
  Construct an empty replay buffer **B** with max memory length l_B.
  for $e \leftarrow 0$ to E training epochs **do**
    Get initial states s_0.
    while s is not the terminal state **do**
      Select option $O_t = \arg\max_o Q^o(S_t, o)$ based on ε-HeuristicExplore.
      Apply attention model to state S_t based on the selected option O_t: $S_t^I = I(S_t, O_t)$.
      Select action $A_t = \arg\max_a Q^a(S_t^I, O_t, a)$ based on ε-HeuristicExplore.
      Execute A_t in simulation to get S_{t+1}.
      $R_{t+1}^o, R_{t+1}^a = HybridReward(S_t, O_t, A_t)$.
      Store transition T into **B**: $T = \{S_t, O_t, A_t, R_{t+1}^o, R_{t+1}^a, S_{t+1}\}$.
    Train the buffer $ReplayBuffer(e)$.
    if e mod $n == 0$ **then**
      Test without action exploration with the weights from training results for n epochs and save the average rewards.

Algorithm 2 Hybrid Reward Mechanism

procedure HYBRIDREWARD()
  Penalize R_t^o and R_t^a for regular step penalties (e.x.: time penalty).
  for δ in sub-goals candidates **do**
    if δ fails **then**
      if option $o_t == \delta$ **then**
        Penalize option reward R_t^o
      else
        Penalize action reward R_t^a
  if task success (all δ success) **then**
    Reward both R_t^o and R_t^a.

Algorithm 3 Hierarchical Prioritized Experience Replay

procedure REPLAYBUFFER(e)
  mini-batch size k, training size N, exponents α and β.
  Sample k transitions for option and action mini-batch:

$$MB^g \sim P^g = \frac{p^{g\alpha}}{\sum_0^{l_B} p_i^{g\alpha}}, \quad g \in \{o, a\}$$

  Compute importance-sampling weights:

$$w^g = \frac{[N \cdot P^g]^{-\beta}}{\max_i w_i^g}, \quad g \in \{o, a\}$$

  Update transition priorities:

$$p^o = \left| Y_t^{Q^o} - Q^o(S_t, O_t | \theta_t^o) \right|$$

$$p^a = \left| Y_t^{Q^a} - Q^a(S_t^I, O_t, A_t | \theta_t^a) \right| - p^o$$

  Adjust the transition priorities: $p^a = p^a - min(p^a)$.
  $\theta_t^g = \theta_t^g + \alpha \frac{\partial L(\theta^g)}{\partial \theta^g}$ according to sample weights w^g, $g \in \{o, a\}$.
  Update target networks: $\theta^{g'} = \theta^g$, $g \in \{o, a\}$.

layers. For the transitions $\{s, o, a, r^o, r^a, s'\}$ stored during the HRL process, the central observation is that if the output of the option-value network o is chosen wrongly due to high error between predicted option-value Q^o and the targeted option-value $r^o + \gamma Q^o(s', o')$, then the success or failure of the corresponding action-value network is inconsequential to the current transition. We therefore propose the Hierarchical Prioritized Experience Replay (HPER) algorithm, in which the priorities at the option-level are based on error directly and the priorities at the lower level are based on the difference between errors coming from two levels. Higher priority is assigned to the action-level experience replay if the corresponding option-level has lower priority. Based on the described approaches, the Hybrid HRL is shown in Algorithms 1, 2 and 3.

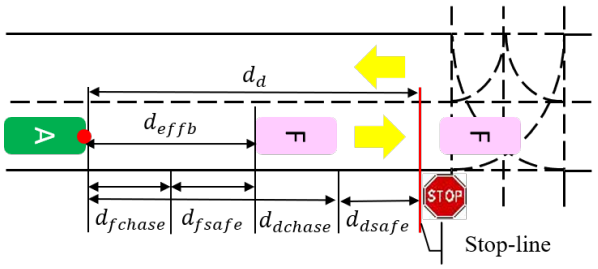

Fig. 4: Autonomous vehicle (green box with A) approaching stop-sign intersection

V. EXPERIMENTS

In this section, we apply the proposed algorithm to the behavior planning of a self-driving car and make comparisons with competing methods.

A. Scenario

We tested our algorithm in MSC's VIRES VTD. We designed a task in which an AV (green box with A) intends to stop at the stop-line behind a random number of front vehicles (pink boxes with F) with random initial positions (see Fig. 4). The two sub-goals in the scenario are designed as *stop at stop-line* (*SSL*) and *follow front vehicle* (*FFV*).

B. Transitions

1) State: $s = \left[v_e, a_e, j_e, d_f, v_f, a_f, d_{fc}, \frac{d_{fc}}{d_{fs}}, d_d, d_{dc}, \frac{d_{dc}}{d_{ds}}\right]$ is the state space where v_e, a_e and j_e are respectively the velocity, acceleration and jerk of the ego car, while d_f and d_d denote the distance from the ego car to the nearest front vehicle and the stop-line, respectively. A safety distance is introduced as a nominal distance behind the target object which can improve safety due to different sub-goals.

$$\begin{aligned} d_{fs} &= \max\left(\frac{v_e^2 - v_f^2}{2a_{max}}, d_0\right), \quad d_{fc} = d_f - d_{fs} \\ d_{ds} &= \frac{v_e^2}{2a_{max}}, \quad d_{dc} = d_d - d_{ds} \end{aligned} \tag{8}$$

Here a_{max} and d_0 denote the ego car's maximum deceleration and minimum allowable distance to the front vehicle, respectively, and d_{fc} and d_{dc} are safe braking distances to the front vehicle and stop-line, respectively.

2) Option and Action: The option network outputs the selected sub-goal: *SSL* or *FFV*. Then, according to the option result, the action network generates the throttle or brake choices.

3) Reward Functions: Assume that for one step, the selected option is denoted as o, $o \in \{d, f\}$. The reward function is given by: **i. For each step**: a. Time penalty: $-\sigma_1$; b. Unsmoothness penalty if jerk is too large: $-\mathbb{I}_{j_e > 1.}\sigma_2$; c. Unsafe penalty: $-\mathbb{I}_{d_{dc}<0}\exp(-\frac{d_{dc}}{d_{ds}}) - \mathbb{I}_{d_{fc}<0}\exp(-\frac{d_{fc}}{d_{fs}})$.

TABLE I: Results comparisons among different behavior policies

	Rewards		Step	Step Penalty		Performance Rate			
	Option Reward r^o	Action Reward r^a		Unsmoothness	Unsafe	Collision	Not Stop	Timeout	Success
Rule 1	-36.82	-9.11	112	0.38	8.05	18%	82%	0%	0%
Rule 2	-28.69	0.33	53	0.32	6.41	89%	0%	0%	11%
Rule 3	26.42	13.62	128	0.54	13.39	31%	0%	0%	69%
Rule 4	40.02	17.20	149	0.58	16.50	14%	0%	0%	86%
Hybrid HRL	43.52	28.87	178	5.32	1.23	0%	7%	0%	93%

TABLE II: Different HRL-based policies

	Hybrid Reward	Hierarchical PER	Attention Model
HRL^0	×	×	×
HRL^1	√	×	×
HRL^2	√	√	×
HRL^3	√	×	√
Hybrid HRL	√	√	√

ii. For the termination conditions: a. Collision penalty: $-\mathbb{I}_{d_f=0.}\sigma_3$; b. Not stop at stop-line penalty: $-\mathbb{I}_{d_d=0.}v_e^2$; c. Timeout: $-\mathbb{I}_{\text{timeout}}d_d^2$; d. Success reward: $\mathbb{I}_{d_d=0.,v_e=0}\sigma_4$ where σ_k are constants. $\mathbb{I}_c$ are indicator functions. $\mathbb{I}_c = 1$ if and only if c is satisfied, otherwise $\mathbb{I}_c = 0$. Assume that for one step, the selected option is denoted as o, $o \in \{d, f\}$ and the unselected option is o^-, $o^- \in \{f, d\}$:

$$
\begin{aligned}
sr &= -\sigma_1 - \mathbb{I}_{\text{timeout}}d_d^2 + \mathbb{I}_{d_d=0.,v_e=0}\sigma_4 \\
r^{option} &= sr - \mathbb{I}_{d_{o^-c}<0}\exp(-\frac{d_{o^-c}}{d_{o^-s}}) - \mathbb{I}_{d_{o^-}=0.}v_e^2 \\
r^{action} &= sr - \mathbb{I}_{j_e>1.}\sigma_2 - \mathbb{I}_{d_{oc}<0}\exp(-\frac{d_{oc}}{d_{os}}) - \mathbb{I}_{d_o=0.}\sigma_3 \\
r^{task} &= sr - \mathbb{I}_{d_{dc}<0}\exp(-\frac{d_{dc}}{d_{ds}}) - \mathbb{I}_{d_{fc}<0}\exp(-\frac{d_{fc}}{d_{fs}}) \\
&\quad - \mathbb{I}_{j_e>1.}\sigma_2 - \mathbb{I}_{d_f=0.}\sigma_3 - \mathbb{I}_{d_d=0.}v_e^2
\end{aligned}
\tag{9}
$$

where sr represents the portion of the reward common to r^{option}, r^{action} and r^{task}.

C. Results

We compare the proposed algorithm with four rule-based methods and some traditional RL algorithms mentioned before. Table I shows the quantitative results for testing the average performance of each algorithm over 100 cases. The competing methods include:

- Rule 1: stick to the option *Follow Front Vehicle* (*FFV*).
- Rule 2: stick to the option *Stop at Stop-line* (*SSL*).
- Rule 3: if $d_d > (d_f + car\_length)$, select *FFV*, w/o *SSL*.
- Rule 4: if $d_f > d_{fc}$, select *FFV*, w/o *SSL*.
- Table II shows the explanations of different HRL-based algorithms whose results are shown in Figure 5.

Fig. 5 compares the Hybrid HRL method with different setups of the proposed HRL algorithms. It shows the average rewards for both task-level rewards, option-level rewards and action-level rewards based on 1000 episodes for each epoch. It shows that the hybrid reward mechanism performs better with the help of HPER. Table II explains the detailed setup for each approach in Fig. 5.

Fig. 6 depicts a typical example case of the relative speed and position of the ego vehicle with respect to the nearest

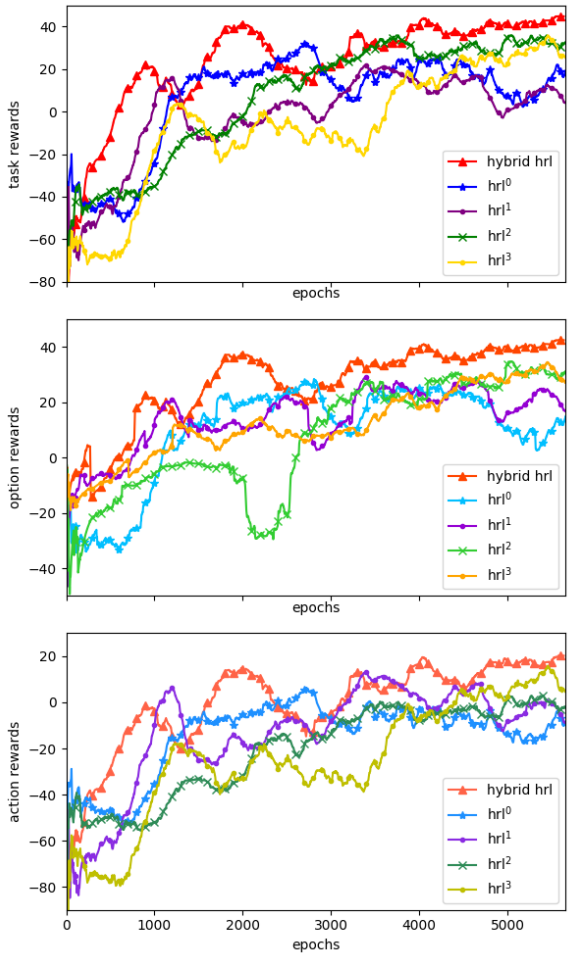

Fig. 5: Training results

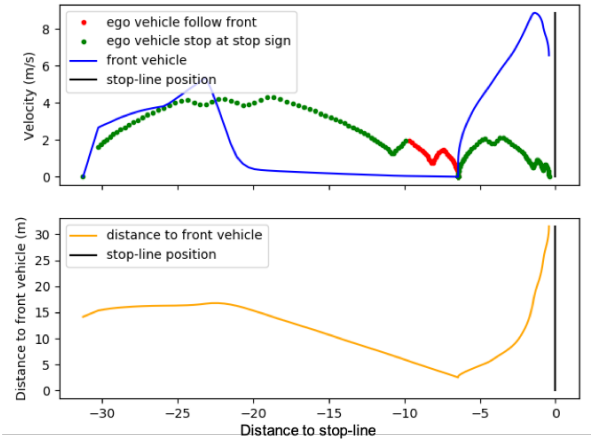

Fig. 6: Velocities of ego car and front vehicles

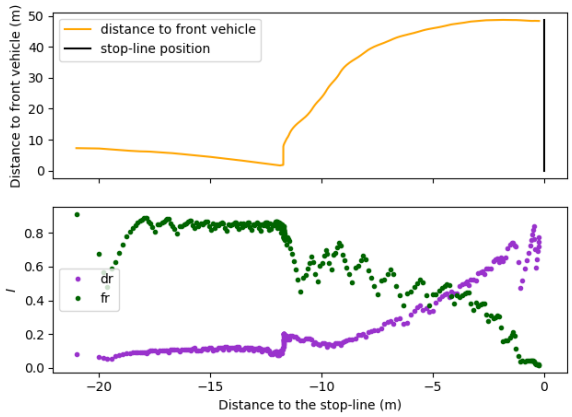

Fig. 7: Attention value extracted from the attention layer in the model. dr and fr are $\frac{d_{dc}}{d_{ds}}$ and $\frac{d_{fc}}{d_{fs}}$ in the introduced state, respectively.

front vehicle as they both approach the stop-line. In the bottom graph we see the ego vehicle will tend to close the distance to the front vehicle until a certain threshold (about 5 meters) before lowering its speed relative to the front vehicle to allow a certain buffer between them. In the top graph we see that during this time the front vehicle begins to slow rapidly for the stop-line at around 25 meters out before taxiing to a stop. Simultaneously, the ego vehicle opts to focus on stopping for the stop-line until it's within a certain threshold of the front vehicle, at which point it will attend to the front vehicle instead. Finally, after a pause the front vehicle accelerates through the stop-line and at this point the ego vehicle immediately begins focusing on the stop sign once again, as desired.

Fig. 7 shows the results extracted from the attention *softmax* layer. Only the two state elements with the highest attentions have been visualized. The upper sub-figure shows the relationship between the distance to the nearest front vehicle (y-axis) and the distance to the stop-line (x-axis). The lower sub-figure is the attention value. When the ego car is approaching the front vehicle, the attention is mainly focused on $\frac{d_{fc}}{d_{fs}}$. When the front vehicle leaves without stopping at the stop-line, the ego car transfers more and more attention to $\frac{d_{dc}}{d_{ds}}$ during the process of approaching the stop-line.

By applying the proposed hybrid HRL, all the option-level and action-level policies can be trained together and the trained-out policy can be separated if the target task only needs to achieve one of the sub-goals.

VI. CONCLUSIONS

In this paper, we proposed three extensions to hierarchical deep reinforcement learning aimed at improving convergence speed, sample efficiency and scalability over traditional RL approaches. Preliminary results suggest our algorithm is a promising candidate for future research as it is able to outperform a suite of hand-engineered rules on a simulated autonomous driving task in which the agent must pursue multiple sub-goals in order to succeed.

REFERENCES

[1] S. Jin, Z.-y. Huang, P.-f. Tao, and D.-h. Wang, "Car-following theory of steady-state traffic flow using time-to-collision," *Journal of Zhejiang University-SCIENCE A*, vol. 12, no. 8, pp. 645–654, 2011.

[2] D. N. Lee, "A theory of visual control of braking based on information about time-to-collision," *Perception*, vol. 5, no. 4, pp. 437–459, 1976.

[3] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.

[4] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[5] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot *et al.*, "Mastering the game of go with deep neural networks and tree search," *nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[6] C. R. Baker and J. M. Dolan, "Traffic interaction in the urban challenge: Putting boss on its best behavior," in *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2008, pp. 1752–1758.

[7] Z. Qiao, K. Muelling, J. M. Dolan, P. Palanisamy, and P. Mudalige, "Automatically generated curriculum based reinforcement learning for autonomous vehicles in urban environment," in *2018 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2018, pp. 1233–1238.

[8] Z. Qiao, K. Muelling, J. Dolan, P. Palanisamy, and P. Mudalige, "Pomdp and hierarchical options mdp with continuous actions for autonomous driving at intersections," in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2018, pp. 2377–2382.

[9] S. Brechtel, T. Gindele, and R. Dillmann, "Probabilistic decision-making under uncertainty for autonomous driving using continuous pomdps," in *17th International IEEE Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2014, pp. 392–399.

[10] D. Isele, A. Cosgun, and K. Fujimura, "Analyzing knowledge transfer in deep q-networks for autonomously handling multiple intersections," *arXiv preprint arXiv:1705.01197*, 2017.

[11] D. Isele, A. Cosgun, K. Subramanian, and K. Fujimura, "Navigating intersections with autonomous vehicles using deep reinforcement learning," *arXiv preprint arXiv:1705.01196*, 2017.

[12] D. Isele, R. Rahimi, A. Cosgun, K. Subramanian, and K. Fujimura, "Navigating occluded intersections with autonomous vehicles using deep reinforcement learning," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 2034–2039.

[13] T. D. Kulkarni, K. Narasimhan, A. Saeedi, and J. Tenenbaum, "Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation," in *Advances in neural information processing systems*, 2016, pp. 3675–3683.

[14] T. G. Dietterich, "The maxq method for hierarchical reinforcement learning." in *ICML*, vol. 98. Citeseer, 1998, pp. 118–126.

[15] N. K. Jong and P. Stone, "Hierarchical model-based reinforcement learning: R-max+ maxq," in *Proceedings of the 25th international conference on Machine learning*. ACM, 2008, pp. 432–439.

[16] R. I. Brafman and M. Tennenholtz, "R-max-a general polynomial time algorithm for near-optimal reinforcement learning," *Journal of Machine Learning Research*, vol. 3, no. Oct, pp. 213–231, 2002.

[17] W. Masson, P. Ranchod, and G. Konidaris, "Reinforcement learning with parameterized actions," in *AAAI*, 2016, pp. 1934–1940.

[18] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," in *Thirtieth AAAI conference on artificial intelligence*, 2016.

[19] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, p. 529, 2015.

[20] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "Prioritized experience replay," *arXiv preprint arXiv:1511.05952*, 2015.

[21] A. Nair, B. McGrew, M. Andrychowicz, W. Zaremba, and P. Abbeel, "Overcoming exploration in reinforcement learning with demonstrations," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 6292–6299.

[22] Y. Gao, H. Xu, J. Lin, F. Yu, S. Levine, and T. Darrell, "Reinforcement learning from imperfect demonstrations," *arXiv preprint arXiv:1802.05313*, 2018.