Guided hierarchical reinforcement learning for safe urban driving

Mohamad Albilani, Amel Bouzeghoub

Guided Hierarchical Reinforcement Learning for Safe Urban Driving

Mohamad Albilani
Samovar, Télécom SudParis
Institut Polytechnique de Paris
Palaiseau, France And Avisto Telecom
mohamad.albilani@avisto.com

Amel Bouzeghoub
Samovar, Télécom SudParis
Institut Polytechnique de Paris
Palaiseau, France
Amel.Bouzeghoub@telecom-sudparis.eu

***Abstract*—Designing a safe decision-making system for end-to-end urban driving is still challenging. Numerous contributions based on Deep Reinforcement Learning (DRL) were developed. However, they all suffer from the cold start issue and require extensive convergence training. Recent solutions for urban driving have emerged based on both Hierarchical Reinforcement Learning (HRL) and imitation learning to overcome these limitations. Nevertheless, they do not guarantee a safe exploration for an autonomous vehicle. In the literature, rule-based systems played a pivotal role in ensuring the safety of self-driving cars, but they require manual rule encoding. This paper introduces GHRL, a decision-making framework for vision-based urban driving that benefits from HRL, and a rule-based system for safe urban driving. The HRL algorithm learns the high-level policies, whereas the low-level policies are guided by the expert demonstration rules modeled with the Answer Set Programming (ASP) formalism. When a critical situation occurs, the system will shift to rely on ASP rules. The state of each policy includes visual features extracted by a convolutional neural network from a monocular camera, information on localization, and waypoints. GHRL is evaluated on the Carla NoCrash benchmark. The results show that by incorporating logical rules, GHRL achieved better performance over state-of-the-art HRL algorithms.**

***Index Terms*—Hierarchical Reinforcement Learning, Self Driving Car, Safe Urban Driving**

I. Introduction

Today, most autonomous vehicle (AV) systems use a hand-crafted modular architecture [4]. However, the modular architecture is criticized for showing poor accuracy in highly interactive environments, such as urban driving. These models are tightly interconnected, which makes them expensive to scale and maintain. These limitations are bypassed by adopting end-to-end architectures, in which a driving policy is learned and generalized without human intervention [4]. The learned driving policy can also be continuously tuned with each driving attempt to achieve human-level performance. A safe driving policy remains an open challenge where the complexity surpasses a few well-defined tasks (e.g., moving box robot). The three main categories of end-to-end AV driving policy are: rule-based methods [16], imitation learning (IL) [5] [4] [6] and reinforcement learning (RL) [7]. The rule-based methods are human-designed predetermined rules structured to achieve the best driving policy by selecting maneuvers and then planning the trajectory [16]. Despite the popularity of rule-based systems, manual rule encoding can put a strain on system engineers as they must anticipate all the crucial and possible rules for each driving scenario [7]. IL-based methods are an effective alternative where the driving policy is learned directly and supervised by mimicking expert demonstrations as training sets. However, these methods require large amounts of labeled training data.

To limit the time-consuming hand-labeled data, solutions that use deep RL (DRL) for end-to-end driver policy learning have been applied to simple driving scenarios, like lane keeping, steering control, and managing the acceleration [15]. However, a primary challenge in DRL lies in guaranteeing the safety of autonomous driving (AD) systems, especially during the exploration phase [32]. In urban driving scenarios, when AVs engage in exploratory behavior, they risk taking actions that could result in catastrophic consequences, potentially jeopardizing passengers' lives. Moreover, DRL typically demands a substantial volume of training data [44]. Also, acquiring such extensive datasets for AD can be exceedingly challenging [33]. These combined difficulties confine the training of AVs primarily to simulation environments and make their transition to real-world driving situations practically unfeasible.

Several distinct approaches exist to achieve safety in DRL [34]. One approach entails restricting the expected cost [35]. Alternatively, using the loss function, another method maximizes the safety constraints [36]. These approaches consolidate safety concerns into a complex loss function, making the optimization more challenging. In contrast, the shielding technique [37] deploys a shield to directly forestall the agent from taking actions that might potentially breach safety regulations during the exploration phase of DRL [38]. However, the shield's strictness may sometimes hinder the learning agent's ability to effectively explore the environment and discover its optimal policy [43].

Recent studies have shown that Hierarchical Reinforcement Learning (HRL) is more suitable for urban driving [17]. HRL helps by breaking the task into smaller sub-tasks with more straightforward state space, thus reducing the required exploration [22]. On the other hand, IL can help with the cold start issue by providing a pre-trained expert policy, which helps guide the agent's actions. Instead of providing the expert's demonstrations as action recommendations, modeling them as

humanlike reasoning using symbolic logical rules increases the information per interaction and does not limit it to the current state [39]. This more informationally-rich interaction method improves the agent's performance compared to existing methods. Combining HRL and IL in urban driving can be a powerful approach to improve the efficiency of the driving task. However, HRL may suffer from the same safety issues as RL, such as exploration of unsafe actions and failure to generalize to new situations [6].

Integrating a rule-based system as a safety mechanism to represent human background knowledge (BK) can be a potential solution to address safety concerns in critical situations when using an HRL approach for urban driving. Guided by the rules, the agent can converge faster by following specific guidelines or constraints during decision-making, reducing the risk of accidents or dangerous maneuvers. Moreover, the rule-based system can provide a fallback option to ensure the safety of the driver, passengers, and other road users. However, it is essential to note that incorporating a rule-based system may limit the agent's adaptability to new situations and may require additional engineering effort to define the rules accurately. Therefore, it is crucial to balance the trade-off between safety and adaptability when designing an HRL-based autonomous driving system.

In that respect, this paper proposes a Guided Hierarchical Reinforcement Learning (GHRL) approach based on vision and localization for end-to-end urban driving. To safeguard the integrity of actions taken during the exploration phase, we inject expert demonstrations expressed in the Answer Set Programming (ASP) [19] formalism into the learning process, guiding the Proximal Policy Optimization (PPO) exploration policy. This fusion of GHRL with ASP offers several noteworthy advantages over alternative methodologies. Firstly, it contrasts the intricacies of calculating complex loss functions, which often exacerbates the optimization process's challenges [34] [35]. Moreover, it circumvents the pitfalls associated with cumbersome shielding techniques, which can lead to exponential complexity in both state and action dimensions. Additionally, unlike methods requiring extensive computational time, our approach offers a more efficient and real-time decision-making process [43].

In dangerous situations, exploration by the agent to learn the best policy might not be feasible and could even result in catastrophic consequences. Therefore, the system relies solely on ASP rules to ensure safety. This approach guarantees the system adheres to predefined safety constraints, preventing potential pedestrian harm. By incorporating a rule-based system in the agent's decision-making process, the agent can benefit from both the exploration of the learned policy and the safety of the rule-based system. Specific situations or events, such as detecting an obstacle or predicting a dangerous situation, trigger the switch from one system to another. We evaluated our method on urban driving scenarios using Carla's simulator [11] and demonstrated its effectiveness in handling various challenges, such as traffic lights and static and dynamic obstacles.

Hence, the main contributions of this paper can be summarized as follows:

- We proposed GHRL, a model-free on-policy RL algorithm. The algorithm employs PPO and is guided by expert demonstration rules expressed in ASP.
- In situations where safety is of utmost importance, the system automatically switches to ASP rules integrated into the agent's decision-making process to take the appropriate action in critical situations.
- We have studied extensive parameters and performed ablation studies on reward shaping.
- The agents can learn efficient driving policies in the CARLA simulator that exhibits a wide range of urban behaviors like lane-following, handling intersections or traffic lights, and avoiding static or dynamic obstacles. The framework is adequately justified using the Carla NoCrash benchmark.

The rest of the paper is organized as follows: Section II discusses the literature review, Section III is dedicated to the contribution of this paper, Section IV details all the experiments and finally, Section V concludes and gives some perspectives.

II. Related Work

Safe end-to-end autonomous driving has been an active research topic in recent years. Most end-to-end systems now fall under one of three paradigms: rule-based, imitation learning, and deep reinforcement learning. Despite much research on end-to-end urban driving, this paper is concerned explicitly with safe end-to-end urban driving. While many previous studies have explored various aspects of urban driving automation, the safety of passengers and other road users must remain a top priority.

A. Rule-based Methods

In [27], the authors presented an AV decision-making system that respects safety considerations and traffic laws following ISO/PAS 21448 [26]. [28] developed a system that imposes the safety of AV by validating the rule priority structure for each decision. [24] proposed a common sense reasoner using ASP for AV end-to-end decision-making by simulating the mind of a human driver. The rule-based system played a pivotal role in ensuring the safety of an AV decision-making system. However, all the possible scenarios must be manually anticipated and encoded. Indeed [27], [26], [28] and [24] presented end-to-end systems while emphasizing safety features. However, such a system relies on a fixed set of rules and needs more flexibility to adapt to unexpected scenarios. Also, this can result in reduced performance and even failure in complex and dynamic driving situations. Additionally, rule-based systems can be challenging to maintain and update as they require manual adjustments for each new scenario.

B. Imitation Learning

IL immediately learns the driving decision model by imitating the expert's demonstrations using supervised learning

techniques. [2] proposed CIL, capable of executing high-level navigation commands and its extension CILRS [13] that introduces a speed prediction head. [5] presented ChauffeurNet that augments the imitation loss to penalize undesirable events in urban driving. [23] proposed LBC by training an agent with privileged information, then using this agent as a monitor for a second agent without access to privileged information. [29] applied DAGGER [30] with essential state sampling to CILRS. The presented approaches effectiveness is restricted as they rely heavily on hand-labeled data, mostly gathered from expert-operated vehicles. The system gathers images and steering angles under different driving conditions with numerous drivers and complex traffic. However, acquiring sufficient naturalistic driving data can be challenging in real-world settings. Additionally, different human drivers may make vastly different decisions in the same situation, which is challenging during training.

C. Reinforcement Learning

Recently, several techniques have advocated using DRL for autonomous urban driving. [15] took a few episodes and a single monocular front-facing camera to employ a pre-trained variational autoencoder (VAE) and encode the visual data with convolutional layers to train a DRL agent to follow a rural road successfully. IARL [14] was the first DRL work capable of completing complex end-to-end urban driving by introducing implicit affordance.

Learning-based methods aim to learn the driving policies without human intervention. However, ensuring their safety is a significant concern. Several works tried to overcome this issue by bypassing the functional requirements of the neural network [3] or verifying the safety of each action in post-hoc methods [10]. In [37], a shield is employed to proactively prevent the agent from taking actions that could potentially result in safety breaches during the exploration phases of both model-based DRL [38] [40] and model-free DRL [41]. This shield is a logical component designed to carefully consider safety constraints, thereby ensuring safety during the exploration of an environment [38]. [41] proposed a shielding technique based on logical neural networks (LNNs) [42] recommends safe actions and avoids unnecessary ones. However, synthesizing an offline shield for discrete-event systems demands an exhaustive, upfront safety analysis for all potential state-action combinations, resulting in exponential complexity in state and action dimensions [37] and becoming over-restrictive [40]. Online shielding lacks worst-case computation time guarantees, potentially allowing the agent to reach the next decision state before the shield determines which action to block. It is suitable in scenarios where alternative actions, like ”waiting,” can be taken if safety analysis is not completed promptly [43].

Some works designed hierarchical methods that learn high-level policies and extract low-level policies by imitating an expert with an additional layer of safety for each option. [9] and [8] proposed HRL with safety constraints. However, it has only been tested on roundabouts and did not include other features of urban driving that we have considered in our study. To overcome these limitations, we propose combining HRL with rule-based systems in this paper. HRL can provide the flexibility and adaptability to navigate complex and dynamic urban environments. In contrast, rule-based systems provide a set of predefined rules that can be applied in critical situations to ensure safety. By combining both approaches, an AV can learn from its experiences in a hierarchical manner and make decisions based on its learned policies. However, in a dangerous or unpredictable situation, the vehicle can switch to a rule-based system to reduce exploration and ensure safety. This hybrid approach can balance exploration and safety, making autonomous urban driving safer and more efficient.

III. Guided Hierarchical Reinforcement Learning

In GHRL, the agent learns to solve the hierarchically structured subtasks, with higher-level policies determining which subtasks to focus on and lower-level policies determining how to solve each subtask. We used the option-critic (OC) framework that provides an end-to-end learning method to construct options [21]. It is a popular approach used in HRL to model decision-making at different levels of abstraction. For high-level policies, options represent actions or subtasks that achieve a specific goal or a subgoal according to waypoints.

A. State

We aim to make the agent's training more efficient and effective in completing the navigation task with dynamic actors by defining the state space S to include sufficient environmental knowledge as features. In our approach, the inputs fed to the VAE consist of all objects detected by Carla's semantic segmentation and traffic light states. The VAE encoder's CNN creates a vector z from a Gaussian distribution. To augment the vector z, we have added external state variables such as waypoint features w. We have also included accurate localization by fusing GPS and IMU [18], speed, and orientation as a matrix m, which predicts additional features to aid training, such as the distance to impact and distance to an incoming event (such as entering an intersection or stopping at a traffic light). Furthermore, augmenting the data variety includes lateral distance and angle with the optimal trajectory. This approach results in a more disentangled agent training process.

B. Reward Shaping

We developed quadratic rewards with a quadratic decrease for each sub-policy. This reward function incentivizes the agent to take actions closer to the optimal one whilst penalizing actions that deviate farther from it. In addition, we integrated speed, deviation, and angle factors into the reward function to evaluate the agent's actions. To ensure that the agent learns effectively, we have demonstrated through various experiments that it is better to multiply the rewards than to add them. For instance, if the agent controls a vehicle, the reward function may reward target speeds and penalize high deviations from the desired path. Similarly, actions that

result in the vehicle maintaining a specific angle or trajectory may also be rewarded. The reward function encourages the vehicle to maintain a target speed, stay centered in the lane, and align with the road. However, it offers high rewards only when several requirements are partially completed. By logically formulating the reward function, the agent can learn more effectively and optimize critical factors such as safety and efficiency. These rewards are defined as follows:

$$R = f(r_v) * f(r_d) * f(r_\alpha) \tag{1}$$

where f is a quadratic function, r_v is the velocity reward function defined by:

$$r_v = \begin{cases} -10 & \text{on infraction} \\ \frac{v}{v_{\min}} & v < v_{\min} \\ 1 & v_{\min} \leq v < v_{\text{target}} \\ \left(1 - \frac{v - v_{\text{target}}}{v_{\max} - v_{\text{target}}}\right) & v \geq v_{\text{target}} \end{cases} \tag{2}$$

where $v_{\min}$ and $v_{\max}$ are the minimum and maximum allowed velocity (speed) according to the law, and v_{target} is the identified target velocity, r_d is the deviation distance reward function defined by:

$$r_d = \begin{cases} 0 & d >= 3 \\ 1 - \frac{d}{d_{\max}} & else \end{cases} \tag{3}$$

where d is the route deviation distance and $d_{\max}$ is the maximum threshold set to 3 meteres. It indicates that r_d decreases with the increase of d. If d is larger than the maximum allowed value, then the agent will get a minimum reward of 0, and the deviation degree reward r_α is calculated as follows:

$$r_\alpha = \begin{cases} 1 - \left|\frac{\alpha_{\text{diff}}}{\alpha_{\max}}\right| & |\alpha_{\text{diff}}| < \alpha_{\max} \\ 0 & \text{else} \end{cases} \tag{4}$$

where $\alpha_{\max}$is the maximum threshold set to 90° and α_{diff} is the angle difference between the vehicle's forward vector and the current way-points forward vector.

C. GHRL Learning

Designing a single end-to-end policy for urban driving with numerous behaviors can be challenging and may lead to poor performance in completing the driving task. To address this issue, we propose an OC framework where high-level and low-level policies are trained synchronously. This framework allows for the incorporation of expert demonstrations by injecting rules into the learning process, which guide the agent's behavior.

1) Low-Level Policies: We train low-level policies using PPO while incorporating expert demonstrations through ASP rules injected by a well-defined hyperparameter p. These rules are considered "on-policy," indicating that the agent generates them during training. By integrating these expert demonstrations into the learning process, the agent can converge faster, reducing the time spent on exploration. Also, this integration allows the agent to benefit from the expert's demonstrations and valuable knowledge for handling various driving scenarios effectively. We represent the set E of expert trajectories as $\tau = (s_0, \mathrm{R}_0, a_0, s_1, \mathrm{R}_1, a_1, ...)$, where each state s_i has a corresponding ASP rule R_i that determines the appropriate action a_i (explained in the following sub-section). Hence, the agent can effectively incorporate the expert's knowledge into decision-making as a pair (s_i,a_i) while having a single source of reward. The new modified policy π_θ^ϕ is used in the clipping function defined in [31] as follows:

$$\pi_\theta^\phi = \begin{cases} \pi_\theta & \text{sampled from Environment with probability } \textit{1-p} \\ \pi_E & \text{sampled from Expert E with probability } p \end{cases} \tag{5}$$

In Algorithm 1, we have modified the PPO by including expert demonstrations in the training data and updating the policy to maximize the probability of taking actions guided by the expert. The hyperparameter p controls the probability of selecting an expert action rather than relying solely on the policy's output, and it can be gradually decreased as the policy improves during training. By utilizing expert demonstrations, we provide the learning algorithm with a set of constraints that steer it toward the desired behavior while allowing for exploring and discovering new approaches. This hybridization enables the algorithm to learn from the expert's knowledge and experiences, leading to more effective decision-making and better overall performance in handling various tasks.

Algorithm 1 GPPO Low-Level Policies

```
Initialize parameters θ, p
Initialize storage ε ← {}
for every update do
    for actor 1,2,...,N do
        Sample τ from expert trajectories E
        if p then
            for steps 1, 2, ..., T do
                s_t, R_t, a_t, r_t, s_{t+1} ~ π_E (a_t | s_t)
                Store transition E ← E ∪ {(s_t, a_t, r_t)}
            end for
        else
            for steps 1, 2, ..., T do
                Execute an action in the environment
                s_t, a_t, r_t, s_{t+1} ~ π_θ (a_t | s_t)
                Store transition E ← E ∪ {(s_t, a_t, r_t)}
            end for
        end if
    end for
    Optimize L^{GPPO} wrt θ, with K epochs and
    minibatches size M ≤ NT
    θ ← θ − η∇_θ L^{GPPO}
    Empty storage E ← {}
end for
```

2) High-Level Policies: The high-level master policy π_{high} is trained after completing all low-policies training. Algorithm 2 is an OC algorithm responsible for learning the different high-level intra-option policies $\pi_{o_t}(a_t \mid s_t, o_t)$ and the termination condition $\beta_{o_t}(s_t, o_t)$ for the option o_t at state s_t. In dangerous situations, φ, the agent will rely solely on the ASP rule set to make safe decisions. This rule set will guide the AV in executing appropriate actions to address the short-term goal, detailed in the upcoming sub-section. The high-level policy over options is a ϵ-greedy form on approximating the option-value function $Q_\Omega(S_t, O_t)$, where Ω is the specific

value function to a particular option o_t at state s_t. OC trains the intra-option policies as follows (see [22] for more details):

$$\frac{\partial L(\theta)}{\partial \theta_\pi} = \mathbb{E}\left[\frac{\partial \log \pi\left(a_t \mid s_t, o_t\right)}{\partial \theta_\pi} Q_U\left(s_t, o_t, a_t\right)\right], \quad (6)$$

where θ_π is the parameter of low-policies, and $Q_U\left(s_t, o_t, a_t\right)$ is the the option-value function. The gradient is calculated as follows:

$$\frac{\partial L(\theta)}{\partial \theta_\beta} = \mathbb{E}\left[-\frac{\partial \beta\left(s_t, o_t\right)}{\partial \theta_\beta}\left(A_\Omega\left(s_t, o_t\right) + \eta\right)\right], \quad (7)$$

where θ_β is the high-level policy termination parameter, and η is the deliberation cost. $A_\Omega(s_t, o_t)$ is the termination advantage. To update the option-value function is as follows:

$$Q_\Omega^{k+1}(s, o) = Q_\Omega^k(s, o) + \alpha\Omega \quad (8)$$

3) Safety Specification: Pre-defined rules are designed to make safe decisions in longitudinal and lateral critical situations φ (explained in the following sub-section). A critical time interval in self-driving cars refers to a situation where the car's autonomous system fails to perceive or appropriately respond to a potential hazard in the surrounding environment, such as a pedestrian crossing the street or another vehicle suddenly changing lanes. During this interval, the ego vehicle c_{ego} will solely apply safety ASP rule-based policy to guarantee safe longitudinal decision-making.

ASP is a declarative programming language for knowledge representation and reasoning that adds negation-as-failure to logic programming. ASP first-order language comprises atoms χ and negative atoms $not\chi$, where $not\chi$ represents negation as failure [20]. Typical normal rules R contain an atom h, the head, and a list of atoms $b_1, ..., b_n$, not $b_{n+1}, ...,$ not b_m,$m, n \geq 0$ as the body. When $m = n = 0$, R is referred to as fact. The basic idea behind ASP is to express a given problem in the form of a logic program, for which we need to search for stable models representing the solutions to the original problem. To obtain a concise expression of the problem, we first use rules in first-order logic. Consequently, the problem will be expressed by a logic program, often referred to as a non-terminal program, containing predicates with variables. To find stable models, the most efficient solvers adopt a two-phase approach. The first phase is the instantiation of the variables, generally referred to as grounding. It involves transforming a logic program expressed in first-order logic into a propositional program. The resulting program will no longer contain any variables but will keep stable patterns identical to the original program. The second phase is the resolution, which calculates the program's stable models of the program.

The environment mapping in the scene is transformed into predicates describing the objects present and their positions. These predicates are represented as facts F in ASP that form the input for the driving decision-making process. F contains, among other information, the speed, the lane, the relative distance, the AV predicted trajectory, lane structure, intersection information, visible traffic signs, lights, and other detected objects. Depending on the facts F, the rules R are applied to produce decisions Y such as accelerating, braking, cruising, changing lanes, and turning left or right. In the following example, the sensors detect a pedestrian while the ego vehicle initiates a right turn. The system should identify this situation as critical, requiring longitudinal and lateral safety measures. Let's consider the following logical program P and $F = \{intent(merge\_right\_lane, 3 : 05), ego\_path(30.215, 3 : 05), obj\_path(Pedst1, 30.215, 3 : 05\}$ a set of atoms:

$$\begin{cases} r1 \text{ :turn\_right\_conditions(T) :- intent(merge\_right\_lane, T).} \\ r2 \text{ :abort\_select\_action(turn\_right, T) :-} \\ \text{ ego\_path(EPath, T),} \\ \text{ obj\_path(Oid, OPath, T),} \\ \text{ path\_intersects(EPath, Oid).} \\ r3 \text{ :path\_intersects(EPath, Oid) :- ego\_path(EPath, T),} \\ \text{obj\_path(Oid,OPath, T'), T=T', EPath = OPath.} \\ r4 \text{ :brake\_conditions(T) :- intent(merge\_right\_lane, T),} \\ \text{path\_intersects(EPath, Oid).} \end{cases} \quad (9)$$

In this scenario, the logical program P defines a set of rules describing the behavior of an AV intending to merge into the right lane at time "T=3:05". These rules consider the presence of pedestrians and intersections to make appropriate decisions. Rule 1 states that the AV's "turn_right_conditions" are met at a time "T" if there is an "intent(merge_right_lane, T)" predicate. This rule implies that the AV's short-term goal is to merge into the right lane, indicating the desired action based on the intent of the navigation system. Rule 2 aborts the action "turn_right" at time "T" under certain conditions. The rule checks explicitly if the ego vehicle's path and the pedestrian's path intersect at a time "T" using the 'path_intersects(EPath, Oid)'predicate. If an intersection is detected, the AV avoids selecting the "turn_right" action to ensure pedestrian safety. Rule 3 defines the predicate 'path_intersects(EPath, Oid)' responsible for checking whether the ego vehicle's path ('EPath') and the pedestrian's path ('Opath')intersect at time "3:05". The rule unifies 'ego_path(EPath, T)' and 'obj_path(Oid, OPath, T')' to determine if the paths intersect, enabling the AV to make informed decisions based on spatial relationships. Lastly, Rule 4 specifies the "brake_conditions" that are met at a time "T" if there is an "intent(merge_right_lane, T)" and the ego vehicle's path and the pedestrian's path intersect at that time. This rule ensures that the AV applies the brakes when necessary to avoid collisions and adhere to the intent to merge into the right lane while ensuring pedestrian safety.

4) Longitudinal Critical Situations: Let c_{ego} and c_{fwd} be two cars such as c_{fwd} is followed by c_{ego} with a distance d. A longitudinal critical situation occurs when c_{fwd} brakes with action $a_{max,brake}$ whilst c_{ego} accelerates with $a_{max,acc}$ then it brakes with $a_{min,brake}$ until a collide with c_{fwd} during a response time τ_{res}. Otherwise, the longitudinal situation is safe. Let t_d be a period during which the situation is critical and $d \leq d_{min}$. The interval $[t_d, t_d + \tau_{res}]$ becomes a critical interval where the ASP safety rules are applied. The mathematical proof of d_{min} computation can be found in [1].

Algorithm 2 Learning High-Level Policies

Initialize external options o_i from low-policies
Initialize master policy π_Ω, option library Ω
Initialize the facts F
Initialize dangerous situations φ
add all pre-trained low policies o_i into Ω
for every update **do**
  if φ **then**
    Execute the logical program (P,F)
  else
    Choose o_t according to s and π_Ω
    Execute o_t according to low-policy π_{o_t} and β_{o_t}
    get s' and R_{t+1}, add (s, o_t, s', R_{t+1}) into buffer
    Update with SMDP Q-Learning
    $Q_\Omega^{k+1}(s,o) = Q_\Omega^k(s,o) + \alpha\Omega$
  end if
end for

5) Lateral Critical Situations: Let c_{ego} be a car driving at a velocity v_{ego} and c a sideways moving car with a velocity v_c during a time interval $[0, \tau_r es]$ distant from each other with a distance d. A lateral critical situation occurs when both cars (or one of them) apply a lateral acceleration $a^{lat}_{max,acc}$ and then brake $a^{lat}_{min,brake}$ until colliding laterally. Otherwise, the lateral situation is safe. The interval $[t^d_{lat}, t]$ is a critical lateral interval time, where t^d_{lat} is a lateral danger threshold time. The mathematical proof of d_{min} can be found in [1].

IV. Experiments

A. Environment Setup and Evaluation Metrics

All the experiments were conducted on the Carla simulator. The environment includes criteria such as high traffic density or complex intersections (i.e., intersections with multiple lanes, merging traffic, and pedestrians crossing). It also contains narrow streets, construction zones, inclement weather, and pedestrian/cyclist interactions, which provide a realistic urban driving experience.

Training Procedure. We evaluated the training phase and the testing outcomes in Town10. The goal of an agent is to complete a trip from point A to point B with no infractions. Points A and B are randomly chosen from a list of 120 points manually placed on the map, with 7140 possibilities. Using the search algorithm A^* [25], a planner calculates the route between both points. Instead of scoring the completion rate of a route, an episode is considered successful when the agent travels a distance of 2500m, for better generalization [12]. To save time, we defined three termination criteria, 1) the agent drives at a speed of $1km/h$ for 5 s, 2) the agent deviates from the center for more than 2.5m, and 3) an agent travels a distance of 2500 m. We used 5 metrics to compare our results; the total rewards, the total distance traveled, the center lane deviation, the angle deviation, and the average speed. [2] suggests using the NoCrash benchmark to evaluate the independent driving policy in various urban settings. This benchmark has three different traffic situations with varying degrees of difficulty: empty, regular (mid numbers of people and cars), and dense (no moving items) (a large number of pedestrians and vehicles). Besides, it specifies 25 routes in Town01 for training and 25 in Town02 for testing, along with six different types of weather. Our autonomous agents are tested in the testing town and the testing weather to see how well they operate.

TABLE I: NoCrash Benchmark

Task	Town	Weather	IARL	LBC	CADER	GHRL
Empty	Train	Train	85	89	95	**97**
Regular			86	87	92	**100**
Dense			63	75	82	**96**
Empty	Train	Test		60	94	**95**
Regular				60	86	**90**
Dense				54	76	**85**
Empty	Test	Train	77	85	92	92
Regular			66	79	78	**88**
Dense			33	53	61	**72**
Empty	Test	Test		36	78	78
Regular				36	72	**85**
Dense				12	52	**65**

Obstacles Avoidance Scenarios. NoCrash benchmark does not consider how the appearance of various cars (such as small cars and big trucks) might affect the agent's behavior. NoCrash benchmark tests the agent over a lengthy path. Each scenario is created using the unpredictable actions of cars and people, for example, pedestrians crossing the road. Furthermore, there are 26 types of pedestrians and 27 types of cars in CARLA 0.9.8. We thus created twenty-six different obstacle avoidance scenarios in Town01 and Town02. Each scenario is a set of short courses to lessen the unpredictability and assess the obstacle avoidance performance and inertia problem [2]. It is worth noting that all the courses are just for testing. We determine the success rates along all paths, just like in the NoCrash benchmark.

Vehicles Avoidance Scenarios. Similarly, we created twenty-seven different obstacle avoidance scenarios. A parked vehicle is produced at a 30-meter distance to block the ego vehicle once it reaches the trigger location. The ego vehicle is required to overtake to avoid a collision.

Pedestrians Avoidance Scenarios. A person appears at a distance of 30 meters on the sidewalk to cross the road. The ego vehicle must halt in time and resume motion after the pedestrian has crossed the street. In the first stage, we gathered a vast and diverse dataset for training our system using Carla's autopilot with additional random noise and 25 training routes under the three criteria specified in the NoCrash benchmark from CARLA.

B. Results On NoCrach Benchmark

We compared our framework with state-of-the-art methods such as CADER, IARL, and LBC. Results of IARL and LBC are taken from CADER. Besides, IARL does not provide test results on testing weather, though we have only the results on the training weather. Table I shows the success rate results on the NoCrash benchmark. GHRL slightly outperformed CADER in empty and regular test weather. Still, it improved 20% in dense traffic and training weather, with 14 and 11 success rates higher than CADER's performance. Our framework carefully completes the urban driving task and performs well in traffic and weather conditions.

TABLE II: Obstacle Avoidance Benchmark

	Vehicle avoidance	Pedestrian avoidance
LBC	55 / 81	73 / 78
IARL	69 / 81	57 / 78
CADER	81 / 81	76 / 78
GHRL	81 / 81	77 / 78

C. Results on Obstacle Avoidance

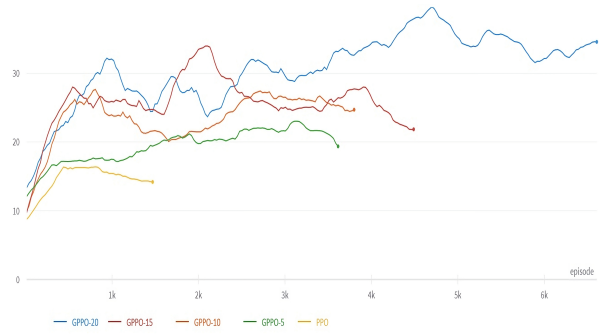
Table II shows the the NoCrash benchmark obstacle avoidance scenarios results. We have executed the evaluation 81 times for vehicle avoidance and 78 times for pedestrian avoidance to be inlined with the tests performed by CADER, LBC, and IARL. Our framework achieved 81 over 81 in vehicle avoidance, the same as CADER, and slightly outperformed it in pedestrians avoidance by achieving 77 over 78.

D. Training Sub-Policies

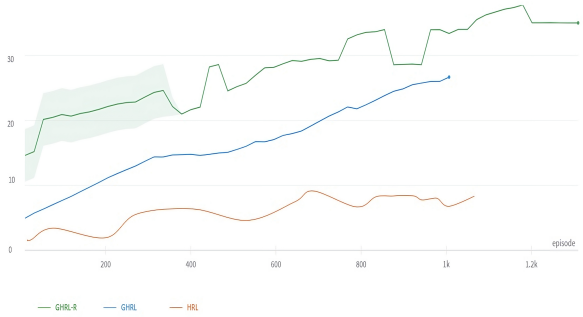
As mentioned previously, the role of all the driving ASP rules is to choose one of the possible actions (turn left, turn right, accelerate, brake) depending on the environment state. For example, the agent cannot accelerate if it is above the speed limit, the traffic light is red, or it is facing an obstacle. [24] provided a list of 35 driving rules, which summarize the total driving ASP rules. Figure 1 (a) depicts the performance of PPO, GPPO-5, GPPO-10, GPPO-15, and GPPO-20 in the case of traffic light management sub-policy. We can see a clear trend of increasing performance as the percentage of rules incorporated into the system increases. Specifically, we observe that GPPO-5, which incorporates 5% rules, performs better than PPO, whilst GPPO-10, GPPO-15, and GPPO-20 incorporate 10%, 15%, and 20% rules, respectively, achieving even better results. This trend suggests that incorporating additional rules into the system can improve its performance, particularly regarding traffic light management. The results indicate that as the percentage of rules increases, the system becomes more efficient at managing traffic lights, leading to better traffic flow and fewer delays. However, it is essential to note that there may be trade-offs between incorporating too many rules and limiting the system's ability to learn and generalize, which can lead to overfitting.

E. Safe High-Policies Experiments

We have evaluated urban driving safety using different RL algorithms with various potential hazards and unpredictable NHTSA (National Highway Traffic Safety Administration) pre-crash scenarios. To evaluate the performance, we used a simulated environment that closely mimics the challenges of urban driving. Specifically, we compared the performance of GHRL with safety rules (GHRL-R), GHRL, and HRL in this environment, focusing on how well they can handle critical situations and accumulate rewards. Fig. 1 (b) sketches the performance comparison results. We can see that GHRL-R outperformed GHRL in critical situations and accumulated more rewards overall. This suggests that the safety features incorporated into GHRL-R allowed it to make better decisions in dangerous situations while achieving high rewards in other scenarios. In contrast, HRL performed poorly compared to GHRL and GHRL-R, achieving lower rewards overall and struggling in critical situations. Such bad performance is due to various factors, such as a need for adequate safety mechanisms or difficulties in learning effective policies.



(a) Learning Sub-policy



(b) Safety Comparison

Fig. 1: GHRL Performances on Carla

V. Conclusion

This paper presents GHRL, a framework that employs a VAE to extract visual features from camera images, localization, and waypoints as navigation input. An RL algorithm is used to learn the high-level policy with OC framework and low-level policies guided by expert demonstration encoded in ASP rules. We also incorporated safety rules into the decision-making process in potentially dangerous situations to ensure that the agent can make safe and responsible decisions, even in complex and challenging situations. We have evaluated GHRL on the Carla NoCrash benchmark and conducted an ablation study to analyze the effect of various network architectures and RL hyperparameters on the proposed framework's performance. The results demonstrate that GHRL outperforms the state-of-the-art methods on Carla's NoCrash benchmark by 20%, achieves four times better than traditional RL, and shows the potential of using HRL for the vision-based control of autonomous vehicles in urban environments. In the upcoming work, we will consider the CARLA leaderboard challenge, a more challenging benchmark than the NoCrash benchmark, as it includes a broader range of traffic scenarios and environmental conditions. This research contributes to the knowledge base of autonomous driving, and future studies can

build on the proposed approach to improve its performance in real-world scenarios.

REFERENCES

[1] Shalev-Shwartz, S., Shammah, S. & Shashua, A. On a formal model of safe and scalable self-driving cars. *ArXiv Preprint ArXiv:1708.06374*. (2017)

[2] Codevilla, F., Müller, M., López, A., Koltun, V. & Dosovitskiy, A. End-to-end driving via conditional imitation learning. *2018 IEEE International Conference On Robotics And Automation (ICRA)*. pp. 4693-4700 (2018)

[3] Filos, A., Tigkas, P., McAllister, R., Rhinehart, N., Levine, S. & Gal, Y. Can autonomous vehicles identify, recover from, and adapt to distribution shifts?. *International Conference On Machine Learning*. pp. 3145-3153 (2020)

[4] Xu, H., Gao, Y., Yu, F. & Darrell, T. End-to-end learning of driving models from large-scale video datasets. *Proceedings Of The IEEE Conference On Computer Vision And Pattern Recognition*. pp. 2174-2182 (2017)

[5] Bansal, M., Krizhevsky, A. & Ogale, A. Chauffeurnet: Learning to drive by imitating the best and synthesizing the worst. *ArXiv Preprint ArXiv:1812.03079*. (2018)

[6] Chen, J., Yuan, B. & Tomizuka, M. Deep imitation learning for autonomous driving in generic urban scenarios with enhanced safety. *2019 IEEE/RSJ International Conference On Intelligent Robots And Systems (IROS)*. pp. 2884-2890 (2019)

[7] Wolf, P., Hubschneider, C., Weber, M., Bauer, A., Härtl, J., Dürr, F. & Zöllner, J. Learning how to drive in a real world simulation with deep q-networks. *2017 IEEE Intelligent Vehicles Symposium (IV)*. pp. 244-250 (2017)

[8] Jamgochian, A., Buehrle, E., Fischer, J. & Kochenderfer, M. SHAIL: Safety-Aware Hierarchical Adversarial Imitation Learning for Autonomous Driving in Urban Environments. *ArXiv Preprint ArXiv:2204.01922*. (2022)

[9] Li, J., Sun, L., Chen, J., Tomizuka, M. & Zhan, W. A safe hierarchical planning framework for complex driving scenarios based on reinforcement learning. *2021 IEEE International Conference On Robotics And Automation (ICRA)*. pp. 2660-2666 (2021)

[10] Krasowski, H., Zhang, Y. & Althoff, M. Safe Reinforcement Learning for Urban Driving using Invariably Safe Braking Sets. *2022 IEEE 25th International Conference On Intelligent Transportation Systems (ITSC)*. pp. 2407-2414 (2022)

[11] Dosovitskiy, A., Ros, G., Codevilla, F., Lopez, A. & Koltun, V. CARLA: An open urban driving simulator. *Conference On Robot Learning*. pp. 1-16 (2017)

[12] Zhao, Y., Wu, K., Xu, Z., Che, Z., Lu, Q., Tang, J. & Liu, C. Cadre: A cascade deep reinforcement learning framework for vision-based autonomous urban driving. *Proceedings Of The AAAI Conference On Artificial Intelligence*. **36**, 3481-3489 (2022)

[13] Codevilla, F., Santana, E., López, A. & Gaidon, A. Exploring the limitations of behavior cloning for autonomous driving. *Proceedings Of The IEEE/CVF International Conference On Computer Vision*. pp. 9329-9338 (2019)

[14] Toromanoff, M., Wirbel, E. & Moutarde, F. End-to-end model-free reinforcement learning for urban driving using implicit affordances. *Proceedings Of The IEEE/CVF Conference On Computer Vision And Pattern Recognition*. pp. 7153-7162 (2020)

[15] Kendall, A., Hawke, J., Janz, D., Mazur, P., Reda, D., Allen, J., Lam, V., Bewley, A. & Shah, A. Learning to drive in a day. *2019 International Conference On Robotics And Automation (ICRA)*. pp. 8248-8254 (2019)

[16] Furda, A. & Vlacic, L. Enabling safe autonomous driving in real-world city traffic using multiple criteria decision making. *IEEE Intelligent Transportation Systems Magazine*. **3**, 4-17 (2011)

[17] Bronstein, E., Palatucci, M., Notz, D., White, B., Kuefler, A., Lu, Y., Paul, S., Nikdel, P., Mougin, P., Chen, H. & Others Hierarchical Model-Based Imitation Learning for Planning in Autonomous Driving. *2022 IEEE/RSJ International Conference On Intelligent Robots And Systems (IROS)*. pp. 8652-8659 (2022)

[18] Albilani, M. & Bouzeghoub, A. Localization of Autonomous Vehicle with low cost sensors. *2022 IEEE 19th International Conference On Mobile Ad Hoc And Smart Systems (MASS)*. pp. 339-345 (2022)

[19] Brewka, G., Eiter, T. & Truszczyński, M. Answer set programming at a glance. *Communications Of The ACM*. **54**, 92-103 (2011)

[20] Clark, K. Negation as failure. *Logic And Data Bases*. pp. 293-322 (1978)

[21] Bacon, P., Harb, J. & Precup, D. The option-critic architecture. *Proceedings Of The AAAI Conference On Artificial Intelligence*. **31** (2017)

[22] Guo, Y., Zhang, Q., Wang, J. & Liu, S. Hierarchical reinforcement learning-based policy switching towards multi-scenarios autonomous driving. *2021 International Joint Conference On Neural Networks (IJCNN)*. pp. 1-8 (2021)

[23] Chen, D., Zhou, B., Koltun, V. & Krähenbühl, P. Learning by cheating. *Conference On Robot Learning*. pp. 66-75 (2020)

[24] Kothawade, S., Khandelwal, V., Basu, K., Wang, H. & Gupta, G. AUTO-DISCERN: autonomous driving using common sense reasoning. *ArXiv Preprint ArXiv:2110.13606*. (2021)

[25] Hart, P., Nilsson, N. & Raphael, B. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions On Systems Science And Cybernetics*. **4**, 100-107 (1968)

[26] Radlak, K., Szczepankiewicz, M., Jones, T. & Serwa, P. Organization of machine learning based product development as per ISO 26262 and ISO/PAS 21448. *2020 IEEE 25th Pacific Rim International Symposium On Dependable Computing (PRDC)*. pp. 110-119 (2020)

[27] Collin, A., Bilka, A., Pendleton, S. & Tebbens, R. Safety of the intended driving behavior using rulebooks. *2020 IEEE Intelligent Vehicles Symposium (IV)*. pp. 136-143 (2020)

[28] Xiao, W., Mehdipour, N., Collin, A., Bin-Nun, A., Frazzoli, E., Tebbens, R. & Belta, C. Rule-based optimal control for autonomous driving. *Proceedings Of The ACM/IEEE 12th International Conference On Cyber-Physical Systems*. pp. 143-154 (2021)

[29] Prakash, A., Behl, A., Ohn-Bar, E., Chitta, K. & Geiger, A. Exploring data aggregation in policy learning for vision-based urban autonomous driving. *Proceedings Of The IEEE/CVF Conference On Computer Vision And Pattern Recognition*. pp. 11763-11773 (2020)

[30] Ross, S., Gordon, G. & Bagnell, D. A reduction of imitation learning and structured prediction to no-regret online learning. *Proceedings Of The Fourteenth International Conference On Artificial Intelligence And Statistics*. pp. 627-635 (2011)

[31] Schulman, J., Wolski, F., Dhariwal, P., Radford, A. & Klimov, O. Proximal policy optimization algorithms. *ArXiv Preprint ArXiv:1707.06347*. (2017)

[32] Gu, S., Yang, L., Du, Y., Chen, G., Walter, F., Wang, J., Yang, Y. & Knoll, A. A review of safe reinforcement learning: Methods, theory and applications. *ArXiv Preprint ArXiv:2205.10330*. (2022)

[33] Kiran, B., Sobh, I., Talpaert, V., Mannion, P., Al Sallab, A., Yogamani, S. & Pérez, P. Deep reinforcement learning for autonomous driving: A survey. *IEEE Transactions On Intelligent Transportation Systems*. **23**, 4909-4926 (2021)

[34] Garcıa, J. & Fernández, F. A comprehensive survey on safe reinforcement learning. *Journal Of Machine Learning Research*. **16**, 1437-1480 (2015)

[35] Achiam, J., Held, D., Tamar, A. & Abbeel, P. Constrained policy optimization. *International Conference On Machine Learning*. pp. 22-31 (2017)

[36] Xu, J., Zhang, Z., Friedman, T., Liang, Y. & Broeck, G. A semantic loss function for deep learning with symbolic knowledge. *International Conference On Machine Learning*. pp. 5502-5511 (2018)

[37] Alshiekh, M., Bloem, R., Ehlers, R., Könighofer, B., Niekum, S. & Topcu, U. Safe reinforcement learning via shielding. *Proceedings Of The AAAI Conference On Artificial Intelligence*. **32** (2018)

[38] Yang, W., Marra, G., Rens, G. & De Raedt, L. Safe Reinforcement Learning via Probabilistic Logic Shields. *ArXiv Preprint ArXiv:2303.03226*. (2023)

[39] Cropper, A., Dumančić, S., Evans, R. & Muggleton, S. Inductive logic programming at 30. *Machine Learning*. pp. 1-26 (2022)

[40] Jansen, N., Könighofer, B., Junges, J., Serban, A. & Bloem, R. Safe reinforcement learning using probabilistic shields. (Dagstuhl: Schloss Dagstuhl,2020)

[41] Kimura, D., Chaudhury, S., Wachi, A., Kohita, R., Munawar, A., Tatsubori, M. & Gray, A. Reinforcement learning with external knowledge by using logical neural networks. *ArXiv Preprint ArXiv:2103.02363*. (2021)

[42] Riegel, R., Gray, A., Luus, F., Khan, N., Makondo, N., Akhalwaya, I., Qian, H., Fagin, R., Barahona, F., Sharma, U. & Others Logical neural networks. *ArXiv Preprint ArXiv:2006.13155*. (2020)

[43] Könighofer, B., Rudolf, J., Palmisano, A., Tappler, M. & Bloem, R. Online shielding for reinforcement learning. *Innovations In Systems And Software Engineering*. pp. 1-16 (2022)

[44] Sutton, R. & Barto, A. Reinforcement learning: An introduction. (MIT press,2018)