Yaw-Guided Imitation Learning for Autonomous Driving in Urban Environments

Yandong Liu†, Chengzhong Xu‡, Hui Kong∗

Abstract—Existing imitation learning methods suffer from low efficiency and generalization ability when facing the road option problem in an urban environment. In this paper, we propose a yaw-guided imitation learning method to improve the road option performance in an end-to-end autonomous driving paradigm in terms of the efficiency of exploiting training samples and adaptability to changing environments. Specifically, the yaw information is provided by the trajectory of the navigation map. Our end-to-end architecture, Yaw-guided Imitation Learning with ResNet34 Attention (YILRatt), integrates the ResNet34 backbone and attention mechanism to obtain an accurate perception. It does not need high-precision maps and realizes fully end-to-end autonomous driving given the yaw information provided by a consumer-level GPS receiver. By analyzing the attention heat maps, we can reveal some causal relationship between decision-making and scene perception, where, in particular, failure cases are caused by erroneous perception. We collect expert experience in the Carla 0.9.11 simulator and improve the benchmark CoRL2017 and NoCrash. Experimental results show that YILRatt has a 26.27% higher success rate than the SOTA CILRS. The code, dataset, benchmark and experimental results can be found at https://github.com/Yandong024/Yaw-guided-IL.git

Index Terms—End-to-end imitation learning, autonomous driving, yaw guidance and attention.

I. INTRODUCTION

Autonomous driving has gained much interest as an essential application of artificial intelligence, from industry to academia [1]. The modular approach, which incorporates perception, localization, planning, and control techniques, is widely used in industry because of its interpretability [2]. In the event of a failure, the module where the defect is located may be analyzed and identified. However, due to the complexity of autonomous driving tasks, the development and maintenance costs of any of the technologies in the module are extremely high [3]. Therefore, an end-to-end imitation learning (IL) approach has recently become a popular research topic in academia [4]. This method learns expert experience through deep neural networks [5]. The environment perceived by the sensors is input into the neural network, and the neural network outputs control variables.

† Center for Cloud Computing, Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, Shenzhen, China. Shenzhen College of Advanced Technology, University of Chinese Academy of Sciences, Shenzhen, China. (e-mail: yd.liu@stat.ac.cn).
‡ The State Key Laboratory of Internet of Things for Smart City (SKL-IOTSC), Department of Computer Science, University of Macau, Macau, (e-mail: czxu@um.edu.mo).
∗ The State Key Laboratory of Internet of Things for Smart City (SKL-IOTSC), Department of Electromechanical Engineering (EME), University of Macau, Macau, (e-mail: huikong@um.edu.mo).

Fig. 1. The CIL selects the road by the end-user’s command. In training, we only use the turning data of lane following to train the model. In testing, images shown in (1)-(3) demonstrate the intermediate trajectory when taking right turn given by an road option command by end-users. However, the vehicle controlled by CIL cannot make a right turning successfully at the intersection.

Imitation learning has been successfully applied to lane keeping (following) and obstacle avoidance by learning driving strategies through extensive human driving experience [6][7]. In a lane keeping scene, there is a single mapping relationship between driving behavior and environment. Therefore, the imitation learning strategy can control vehicles to turn left, turn right and go straight. However, in the urban environments, road option become a major obstacle to applying imitation learning. Road option, that is, when a vehicle is at an intersection, T-shape junction or roundabout, the vehicle needs to decide whether to turn left, right or go straight, as shown in Figure 1. In such cases, scene and vehicle behavior cannot be modeled by a single mapping. As a result, vehicles cannot select a direction only by sensing the surrounding environment using a camera or LiDAR, and distinguishing the turn of lane keeping from that of road option is a big challenge in applying imitation learning in urban environments.

To deal with this issue, the conditional imitation learning (CIL) method [8] exploits human commands to select the corresponding branch network by executing a command of an end-user, where each individual branch network of CIL is trained based on a large amount of image-control pairs sampled accordingly. Generally, when the distribution of sampled training data is unbalanced, the branch network parameters cannot be well optimized for the specific road option that lacks enough training samples. As shown in the example in Figure 1, we use the turning data of lane keeping to train the model and apply it to the case of the right
turning of road option. The vehicle cannot complete the task at the intersection during the test. From this example, we observe that the model trained from the right-turning data of lane following cannot be adapted well to the road option branch. In another word, the CIL does not have enough generalization ability.

Besides, the complexity of scenes is another major challenge for applying imitation learning in urban environments. Due to weather, traffic signals, and dynamic obstacles, urban environments are extremely complicated [9]. Therefore, a good perception module is needed to provide accurate scene representation and feature extraction for the decision network. Especially, it would be more valuable if the learned features are helpful to reveal the causal relationship between scenes and decisions.

In this paper, we propose a fully end-to-end imitation learning architecture for autonomous driving, Yaw-guided Imitation Learning with ResNet34 Attention (YILRatt), where the yaw information is derived from the planned trajectory information instead of the end-user’s command. Due to the yaw guidance, our method has more powerful adaptation ability, and can generalize well to both the turning cases of road option and lane-following scenarios. In this sense, the method improves efficiency of data utilization (the experimental part Section IV-A for details). The network input is an RGB-image captured at each moment. The perception network uses ResNet34 as the backbone, and the attention mechanism strengthens the image feature area, weakens the chaotic area, and enhances the perception ability. Trajectory yaw and vehicle speed are used as measurement inputs to the fully connected network. YILRatt has achieved a 26.27% higher success rate than Conditional Imitation Learning ResNet (CILRS), testing on improved benchmark CoRL2017 and NoCrash, respectively. By analyzing the attention heat maps, we can reveal some causal relationship between decision-making (steer, throttle, and brake) and scene perception, where, in particular, failure cases are caused by erroneous perception.

II. RELATED WORK

Standard imitation learning obtains the control strategy through collected expert experience. Usually, the expert data set is composed of observation-behavior pairs \((o_t, a_t)\) at time \(t\). Similar to the training process of supervised learning, the observation data are the input of the network \(N(o; \theta)\), and the behavior data are the labels. Network parameters are optimized by minimizing the loss function (1) of prediction and expert behavior.

\[
\min_{\theta} \sum_i \mathcal{L}(N(o; \theta), a_i)
\]

However, imitation learning assumes that environmental observations and behaviors in expert data satisfy a single mapping relationship, i.e., \(a_i = E(o_i)\). When enough data are collected, supervised learning can be used to obtain an approximate function of the expert strategy. For example, imitation learning is successful in tasks such as lane keeping and obstacle avoidance given enough collected data. However, in urban scenes, unmanned vehicles face the task of non-single mappings. The expert behavior at this time is not only determined by environmental observations but also related to the destination’s location, i.e., \(a_i = E(o_i, y_i)\), where \(y_i\) is a vector related to road option. We choose the yaw information, which is derived from planned trajectory, just in front of the vehicle as the \(y_i\). Therefore, the collected data becomes \(D = \{(o_i, y_i, a_i)\}^N_{i=1}\). The objective function is adjusted to (2).

\[
\min_{\theta} \sum_i \mathcal{L}(N(o, y_i; \theta), a_i)
\]

The very early work of imitation learning [10] used a three-layer network to learn human driving behavior to accomplish lane keeping. In recent years, with the development of deep learning, models have stronger environmental perception ability. Imitation learning regained a new life. Especially, [6] applies imitation learning to real-world self-driving scenarios. Primitive imitation learning network architectures can only handle single mapping scenarios. Conditional Imitation Learning (CIL) uses branching networks to solve the problem of non-single mapping at the intersection [8]. The design of branching networks allows human and unmanned vehicles to interact. However, as shown in Figure 1, different road option data can only train corresponding model branches, which leads to a poor generalization ability and low data-utilization efficiency. GPS coordinate guidance offers another way of road option [11]. The unprocessed GPS coordinate information cannot provide a better representation for the planned route, which increases the difficulty of learning. In response to the shortcomings of the above two methods, we propose trajectory yaw guidance for the road option, where the yaw information derived from the trajectory waypoints is exploited as the input to the network to guide the vehicle to select the correct road.

To improve the end-to-end imitation learning control accuracy, researchers have done a lot of work on both the perception module and the affordance information. Using only RGB images as input for the perception network, CILRS [9] improves the accuracy of perception in complex scenes by using ResNet34’s strong feature extraction capability. [11] creates a multi-camera system that can offer data for a 360-degree view of the vehicle’s surroundings. Furthermore, [12] uses semantic segmentation technology to improve the perception of the environment. [13] uses semantic segmentation technology while adding geometry and motion with computer vision. Multi-sensor fusion technology is an important method to compensate for the shortcomings of a single sensor. [14] fuses RGB-image with the depth information provided by lidar. The analysis shows that multimodal perception is better than single modality. [15] also uses RGB-camera and lidar, and network parameters are optimized by a loss function including the semantic segmentation result. On the other hand, the affordance approach obtains the environmental features directly through vehicle-road cooperation and human-computer interaction. Conditional Af-
fordance Learning (CAL) [16] provides more effective high-dimensional information for network by adding affordances such as traffic lights’ status and the vehicle’s distance from the lane centerline. Advice from passengers is used as input information to the network [17]. The method provides a way for passengers to interact with unmanned vehicles. At the same time, the control strategy in complex scenarios (to avoid pedestrians traffic accidents) is optimized. The bird-view image provides the road and other vehicle information around the unmanned vehicle to optimize the strategy [18].

Imitation learning is based on expert experience data to train a model. Therefore, the amount and distribution of collected data become the strategy bottleneck. Various urban scenes are designed to provide rich data sources for imitation learning [19]. Data aggregation technology improves the generalization ability of the model [20]. Adding perturbations to the expert experience and penalizing fatal scenarios with a loss function increases the robustness of the model [21]. We use the Carla simulator to collect about 200 thousand expert data to train the model. Furthermore, we analyze the distribution of data based on road option and weather. At last, in the Section IV-A, experiments have shown that the model based on yaw guidance has strong generalization ability under extreme data distribution conditions.

III. YILRatt Architecture

![YILRatt Architecture](image)

**Fig. 2.** Our proposed architecture, called YILRatt, consists of perception, speed, and action modules. The perception module combines ResNet34 with a linear attention mechanism to extract environmental features from a single frame of RGB image. The action module fuses features, measurements and trajectory yaws to predict the action. Speed module improves the accuracy of longitudinal prediction.

The end-to-end imitation learning architecture YILRatt consists of perception \( P \), action \( A \), and speed \( S \) modules, as shown in Figure 2. RGB images \( o_i \) taken by the onboard camera taken at each time instance is input into the perception module. The output of the module is the environmental feature representation \( P(o_i) \). The vehicle speed \( s \) is obtained by the Carla API. The yaw \( y \) is calculated by the global trajectory. The action module predicts the behavior of the vehicle by concatenating the \( P(o_i) \), \( s \) and \( y \). The behavior includes \( steer \in [-1, 1] \), \( throttle \in [0, 1] \), and \( brake \in [0, 1] \). The speed module minimizes the error between the predicted speed and the vehicle speed by supervised learning. The loss of the speed module is added to the total loss function (3) to improve the accuracy of longitudinal prediction. We obtain an expert approximate strategy by optimizing the YILRatt’s parameter \( \theta_P, \theta_A, \theta_S \).

\[
\alpha \cdot \mathcal{L}(A(s_i, y_i; P(o_i; \theta_P); \theta_A)), a_i) + \\
\beta \cdot \mathcal{L}(S(P(o_i; \theta_P); \theta_S), s_i)
\]

**A. Yaw Guidance**

![Image of Yaw Guidance](image)

**Fig. 3.** Through the global planning of the trajectory, the yaw angle is calculated. The vehicle is guided to select the correct road and drives to the destination.

An assumption of the method is that the destination is known. Moreover, the global trajectory is planned to ensure that the unmanned vehicle obtains the yaw in the road option. In an urban scenario, it is a reasonable assumption for an end-to-end method for autonomous driving [8]. The guidance information can be either the coordinates or the yaw of the trajectory. [11] uses the absolute coordinates of the vehicle as guidance information, which results in a location-specific vehicle control network instead of a road-option (direction choice) one. This means that the network learned by using specific-scene training data can be applied to the same scene in general. Therefore, using absolute GPS coordinates as guidance will result in poor generalization ability in road option. In contrast, the guidance yaw information in our method is a relative quantity, and the yaw guidance can map left (right) turns to the same control amount even in different quadrants (Figure 3), where the left turn (right turn) of different scenarios has the same guidance information. As a result, our method has a more generalization power than the method using absolute quantities as guidance information, e.g. [11]. In another words, our method is more efficient in data utilization where we can achieve better end-to-end autonomous driving performance with much fewer training samples. Experiments in Section IV-A can show that yaw guidance is more accurate, and it is easier for our road option network to select the road direction successfully.
By Carla API, we can get yaw from trajectory waypoints. To satisfy the needs of the real world applications, we obtain the yaw information based on the trajectory coordinates. The coordinate difference between two adjacent waypoints on the trajectory is calculated, and then its arc tangent is calculated and set as the yaw angle in Figure 3. $T_{yaw}$ is a vector composed of yaw information of multiple consecutive waypoints. Thus, the yaw guidance, $y = T_{yaw} - yaw_1$, for road option can be obtained. The guidance method not only achieves the road option, but also improves the model’s generalization ability. The model trained on the steering data of the lane-following scenarios can be applied to the steering scenario of the road option, and vice versa. For example, with the same setting as the failure case of conditional imitation learning in Figure 1, Figure 3 (1)-(3) show that our yaw-guided imitation learning successfully completes the right turn.

### B. Perception Module

We use the ResNet34 as the backbone of the perception module. Using ResNets, the gradient can flow directly from the back layer to the initial filter through the jump connection. The disappearance of the gradient of the deep network is solved, thereby improving the accuracy of the image recognition. However, the visual reasoning of ResNets in environment understanding is largely difficult to understand, hindering the understanding of success and failure. Especially in application scenarios with demanding safety requirements such as autonomous driving, we need to understand the causal relationship between decisions and scenarios. Therefore, we introduce the attention mechanism to explain the causal relationship by attention heat maps.

The core idea of the attention in this paper is to combine the local features of the middle layer and the global features of the output layer to strengthen the salient area and suppress the information chaotic area [22]. ResNet34 has the local feature $F^l = \{f^l_1, f^l_2, \ldots, f^l_n\}$ features in the layer $l \in \{1, 2, \ldots, L\}$. $f^l_i$ is the output vector at spatial location $i$ of $n$ total spatial locations. The global feature vector of ResNet34 is $g$, $f^l_i$, and $g$ are two arguments of the same dimension of the compatible scoring function (4).

$$C_i = \langle f^l_i + g, \theta_{att} \rangle, \quad i \in \{1, \cdots, n\} \quad (4)$$

We can simplify the two arguments to an addition operation by element-wise. $\theta_{att}$ can be trained to obtain corresponding features related to the driving scene. The output of the compatible function is the set of scores $C(\hat{F}^l, g) = \{c_1, c_2, \ldots, c_n\}$, where $\hat{F}^l$ is the feature of $F^l$ under a linear mapping of the $f^l_i$ to the dimensionality of $g$. After being normalized by softmax, the score $c^l_n$ is used as the weight of the local feature. The results of different layers are concatenated to provide environmental features for the action and speed network.

### IV. EXPERIMENTS

In this section, we present the experimental setup, results and analysis. Firstly, we compare the Yaw-guided imitation learning with other road option methods in a small-scale static “square” scene. The success rate of turning shows that the data utilization of the Yaw-guided method is high. Next, in the urban environment Carla Town01 and Town02, we compare different architectures in benchmark. Then, by analyzing the experimental metrics such as success rate, lane violation and traffic light violation, we show that YILRatt is the SOTA end-to-end imitation learning architecture. Finally, the causes of failure cases are analyzed by attention heat maps.

#### A. Yaw-guided Road Option in the Static “Square” Scene

![Fig. 4. Comparison trajectory of three road option methods (XY-guided IL, Yaw-guided IL, CIL) and standard Trajectory. (a) and (b) are local and global bird’s-eye views of the test scene. (c) is the view from onboard camera, that is, the single RGB-image inputs into the network. (d) are trajectory details of the three turning tasks.](image)

![Fig. 5. Bar charts show the distribution of three training datasets. Subset (1) only contains the turning data of road option; Subset (2) only contains the turning data of lane keeping; The Full dataset is the union of two subsets.](image)

1) Experimental Setup: We compare the road option methods (XY-guided Imitation Learning, Yaw-guided Imitation Learning, Conditional Imitation Learning) in the
“square” scene including the left and right turns of the intersection, T-junction and lane-keeping, as shown in Figure 4. There are many factors that affect the strategy, such as the initial value of the model parameters, the sampling order of the mini-batch. It is beyond the scope of this paper, there are detailed experimental analysis in [9]. We consider the influence of data distribution on the strategy. We collect three different training datasets in the “square” scene of the Carla simulator including two extreme situations. Data distribution is shown in the bar of Figure 5.

- Subset (1): only the turning data of the road option.
- Subset (2): only the turning data of the lane keeping.
- Full dataset: union of the two subsets.

We use three datasets to train different road option models. Then, 32 turning tasks are tested by the models, as shown by the gray line in Figure 4 (a). And we count the success rate. Further, to verify the robustness of the methods, we add Gaussian noise to the models that 100% complete the tasks and test again.

| Dataset      | XY-guided | Yaw-guided | CIL   | Carla-yaw IL |
|--------------|-----------|------------|-------|--------------|
| Subset (1)   | 22        | 29         | 18    | 29           |
| Subset (2)   | 9         | 12         | 8     | 11           |
| Full dataset | 26        | 32         | 32    | 32           |

* To compare with yaw calculated by coordinate, yaw obtains directly from the Carla API.

2) Experimental Results and Analysis: The success rate is shown in Table I. To compare with the Yaw-guided IL obtaining yaw by coordinate calculation, the Carla-yaw IL is added to the experiment, which obtains the yaw directly by Carla API. On the subset, the guided method has more successful times than the CIL. Especially, the models trained by Subset (1) are far better than those trained by Subset (2). The Yaw-guided IL has more successful times than the XY-guided IL. Therefore, we consider that: (1) the guided method has generalization ability when facing the situation of the road option, but the CIL does not. (2) Compared with the turning task of lane keeping, the turning task of road option is more difficult. (3) Compared with coordinates, the yaw guidance has a higher success rate. To further show point of view, we draw the trajectory of specific tasks in the Carla simulator. In Figure 4 (c) “Turn left in lane-follow”. We use Subset (1) to train the model and test the left-turning task of lane following. The guided methods complete the task. On the contrary, CIL lacks the training data of lane following, and the corresponding branch network parameters cannot be optimized, resulting in failure. In the “Turn right in the T-junction”, we use Subset (2) to train the model and test the right-turning of road option. Only Yaw-guided IL completes the task. XY-guided IL and CIL cannot generate the right-turning strategy by learning in the lane-following data, which leads to failure. Therefore, the Yaw guided IL is the best road option method and the turning of the road option is more difficult than the lane-keeping turning.

On the full dataset, both Yaw-guided IL and CIL complete all tasks. To compare the quality of task completion, we present a specific trajectory, in Figure 4 (d) “Turn left in the intersection”. We use the full dataset to train models and test the left-turning task of road option. Yaw-guided IL and CIL complete the task, and the trajectory of CIL is closer to the standard trajectory. Therefore, compared with CIL, the guided method is more sensitive to the data distribution of left and right turns. XY-guided IL not only fails the task but also moves in the opposite direction. Because the guided data, as the input of the network, determines the behavior of the vehicle together with the image data. Obviously, the image data has a greater impact on vehicle behavior at this moment.

| Gaussian noise | N(0, 10^2) | N(0, 20^2) | N(0, 30^2) |
|---------------|------------|------------|------------|
| Yaw-guidance  | 31.6 ± 0.49| 30.4 ± 0.80| 20.0 ± 1.26|
| Carla-yaw     | 31.2 ± 0.40| 30.8 ± 0.75| 18.4 ± 1.02|

In the real world, GPS navigation signals are disturbed by noise (for the guided IL), and users maybe issue wrong instructions (for the CIL). Therefore, the robustness of the road option method is very important for applying the model. We add noise to the models, Carla-yaw IL, yaw-guided IL and CIL, which complete all tasks. The guided method inputs the trajectory data into the network. Therefore, we directly add Gaussian noise to the raw guided data. Unlike the guided method, CIL selects different branch conditional networks by command. Therefore, we add five consecutive frames of wrong commands as noise based on the different probabilities. We choose the five different random numbers to repeat the experiments and calculate the mean and standard deviation of success times. The experimental results are shown in Table II. Compared with the guided method, CIL is significantly affected by noise. Especially, when the wrong command is added with 10% probability, the variance of the experimental results reaches 10.29^2. With the increase of noise, the successful number of the guided methods gradually decreases, but the variance does not change much. Therefore, the guided method is more robust than the CIL.

B. YILRatt in Urban Environments

We set up experiments in Carla urban environments (dynamic obstacles). We train the YILRatt and CIL, CILRS models in the Town01 and test in the new scene Town02. Benchmark testing demonstrates the superior performance of the YILRatt architecture.

1) Experimental Setup: Dataset is the basis to ensure that the model completes the navigation task. We collect about 200 thousand data under four weather conditions (ClearNoon, WetNoon, HardRainNoon, and ClearSunset) in Carla Town01, as shown in Figure 6 (a). The unmanned
vehicle is controlled by the Carla AI algorithm to collect data. Control noise is added to 10% of the data to improve the robustness of the model. But the recorded data is still the control signal. The dataset is divided into two parts: RGB images and labels. The size of the original image collected by the onboard camera is $3 \times 600 \times 800$ (channel, height, width). After cropping and scaling, the image size change to $3 \times 88 \times 200$. This size reduces the amount of calculation while ensuring that all environmental information is included. Labels include control (steer, throttle, and brake), road option (straight, left, right, and lane-follow), yaw and speed. The dataset is evenly distributed under the four weather conditions. However, lane keeping has far more data than the others. Because it has a high probability of appearing in the simulation scene, and contains turning data. The steer obeys the normal distribution. Throttle and brake are mutually exclusive, as shown in Figure 7.

So far, CoRL2017 and NoCrash are two commonly used benchmarks based on Carla 0.8.4. Refer to the original design standards, we have made improvements in the latest Carla 0.9.11. Benchmark is divided into two types of scenarios: Training condition and New Town02 & Weather (WetCloudyNoon, SoftRainSunset). Navigation tasks are divided into five categories according to the degree of difficulty: Straight, One Turn, Navigation Empty, Nav. Regular (Vehicles: 30, Pedestrians: 10), Nav. Dense (Vehicles: 50, Pedestrians: 30). Each type of navigation task contains 25 paths, of which the route distance in Town01 is not less than 1.0 km, and the distance in Town02 is not less than 0.3 km. We use three metrics to measure the strategy: success rate, lane violations, and traffic light violations. We consider that the task is successful when the vehicle arrives at the destination without collision within the specified time. Especially, the violations of failed tasks are not included in the statistics.

All models are trained using the above dataset. The models use Adam optimizer with mini-batch 512 samples. The initial learning rate is 0.001. And every 10 epochs, the learning rate is reduced by half. The training process includes 100 epochs. The RGB image, speed and yaw are normalized processes and as inputs to the model. Image augmentation (blur, noise, brightness) are used to improve the generalization ability of the model. The outputs of the model are control and speed. The loss function is composed of the two parts and its weights are $\alpha = 1.0$ and $\beta = 0.1$ respectively. We record the model parameters that minimize the loss of the evaluation in 100 epochs and use them for testing the benchmark.

2) Benchmark Testing: We test four methods, the CIL, CILRS, YILRatt and the pretrained model by Pytorch ResNet34, on the benchmark. These end-to-end imitation learning methods only use a fixed dataset for training. No additional auxiliary information is required. We show the experimental results in Table III. The result shows the navigation success rates of the four algorithms in static and dynamic scenarios. Static scenes include Straight, One Turn, Navigation Empty, Navigation regular and dense conditions, the success rate of YILRatt obviously improves the generalization of the model. In the regular and dense conditions, the success rate of YILRatt (pretrained) is increased by $52\%$ and $48.67\%$ respectively than that of CILRS. The experimental results also show that the pretrained model is helpful to strategy optimization, but it is not obvious.

We count the number of lane and red traffic light violations in Table IV. We do statistics under the New Town02 & Weather dynamic conditions. Mean and standard are

---

**Fig. 6.** The Carla urban environments. The data is collected in Town01 for training. Town02 is for testing. Views from onboard camera are based on the different weather conditions.

**Fig. 7.** The figure shows the distribution of training data under four weather conditions. The data is uniformly distributed under the road option (left, right and straight). The lane-follow data is about 7 times of others. Because the lane-follow data includes not only straight but also turns. The data distribution of steer, throttle, and brake are displayed in turn.
### TABLE III
**Navigation Success Rate**

| Task                | CIL  | CILRS | YILRatt | YILRatt (pretrained) | CIL  | New Town02 & Weather | CILRS | YILRatt | YILRatt (pretrained) |
|---------------------|------|-------|---------|----------------------|------|----------------------|-------|---------|----------------------|
| Straight            | 97   | 98    | 99      | 97                   | 100  | 100                  | 100   | 100     | 100                  |
| One Turn            | 95   | 100   | 99      | 100                  | 100  | 100                  | 100   | 100     | 100                  |
| Navigation (Empty)  | 41   | 50    | 92      | 86                   | 64   | 44                   | 88    | 94      | 100                  |
| Nav. (Regular)      | 67   | 33    | ±2.49   | 66.67 ± 1.70         | 41.33| ±4.92               | 48.00 ± 16.27 | 92.33 ± 2.05| 92.0 ± 2.16 | 46.67 ± 1.89 | 35.33 ± 6.18 | 76.66 ± 9.94 | 87.33 ± 6.80 |
| Nav. (Dense)        | 55   | 33    | ±2.49   | 63.67 ± 1.70         | 41.33| ±2.49               | 66.67 ± 1.70 | 93.67 ± 2.62| 25.33 ± 5.00 | 34.0 ± 4.32 | 76.0 ± 3.27 | 82.67 ± 0.94 |

### TABLE IV
**Lane and Traffic Light Violations**

| Task                     | CIL  | New Town02 & Weather | CILRS | YILRatt | YILRatt (pretrained) |
|--------------------------|------|----------------------|-------|---------|----------------------|
| Lane Violation (Infraction per km) | Nav. (Regular) | 4.70 ± 0.22 | 4.65 ± 0.30 | 3.23 ± 0.36 | 2.91 ± 0.93 |
|                          | Nav. (Dense) | 4.53 ± 0.54 | 4.42 ± 0.39 | 3.38 ± 0.04 | 1.94 ± 0.01 |
| Traffic Light Violation (Violations/the total number) | Nav. (Regular) | 0.50 ± 0.03 | 0.52 ± 0.08 | 0.31 ± 0.04 | 0.36 ± 0.04 |
|                          | Nav. (Dense) | 0.63 ± 0.27 | 0.35 ± 0.14 | 0.34 ± 0.07 | 0.35 ± 0.10 |

---

**Fig. 8.** The attention map is derived from the YILRatt(pretrained). Four sets of images focus sharply on the objects obstructing the movement of vehicles.

**Fig. 9.** Heat maps are four consecutive frames. It demonstrates the process of the vehicle avoiding a pedestrian, slowing down, and then accelerating.
derived from the three runs. YILRatt (pretrained) has the lowest number of lane violations (2.91 ± 0.93 in Regular and 1.94 ± 0.01 in Dense). And in terms of signal light recognition, YILRatt performs best, (0.31 ± 0.04 in Regular and 0.34±0.07 in Dense). The possible reason why YILRatt is better than pretrained model is: ImageNet dataset is used to pretrain ResNet34, which has no obvious effect on traffic lights recognition. Experimental data shows that the end-to-end imitation learning method we proposed can perform best not only to complete the task quantity but also in quality. Therefore, the YILRatt can well integrate the perception and control modules to realize end-to-end autonomous driving.

3) Attention Heat Maps: On the one hand, the Attention module improves the perception ability of the model by fusing local and global features. On the other hand, we use the attention heat maps to understand the causal relationship between the scene and decision. Especially, maps help us analyze the reason of failure cases. We superimpose the feature map after attention transform with the raw image with weight value to get the heat map. We select the pedestrian, the vehicle, the traffic light, and fire hydrant to demonstrate in Figure 8. The heat maps of Layer 1 pay attention to the passable area of the vehicle. The heat maps of Layer 2 pay attention to lane line characteristics. The heat maps of Layer 3 pay attention to details. The vehicle behavior corresponding to the scene is braking. However, in the hydrant case, there was an unpredictable result. The vehicle mistakes the fire hydrant for an obstacle to avoid and makes a decision to stop, which leads to the failure of the task.

Furthermore, we analyze the scene of the vehicle avoiding a pedestrian through sequential frame heat maps in Figure 9. When the pedestrian is far away from the vehicle, the vehicle pays attention to the nearby passable area and maintains the speed (throttle=0.41). When a pedestrian obstructs the vehicle, the vehicle pays attention to the pedestrian and brakes (brake=0.99). After the pedestrian passes, the vehicle again shifts its attention to the passable area and accelerates (throttle=0.69). By attention heat maps, we visualize environmental characteristics and clarify the driving behavior of the unmanned vehicle. For more case studies, please see the supplementary video https://youtu.be/EQg3ZPHTf48.

V. CONCLUSION

The experimental data collection, benchmark testing, and result analysis show that our designed end-to-end imitation learning architecture YILRatt was better than the SOTA CILRS. The yaw-guided method achieves road option and improves data utilization. The perception module based on ResNet34, fusing attention mechanism improves the accuracy of scene recognition. Attention heat maps explain the reason for making decisions.

Nonetheless, from the analysis of failure cases, it can be seen that the YILRatt cannot handle sequence information. Models with timing processing capabilities, such as Transformer, can be considered. What follows is the increase of model parameters, we need to collect more data to train the model to prevent overfitting. Another problem is that the IL has an expert strategy bottleneck. Reinforcement learning is a good attempt to explore better strategies.

REFERENCES

[1] E. Yurtsever, J. Lambert, A. Carballo, and K. Takeda, “A survey of autonomous driving: Common practices and emerging technologies,” IEEE Access, vol. 8, pp. 58 443–58 469, 2020.
[2] Élői Zablocki, H. Ben-Younes, P. Pérez, and M. Cord, “Explainability of vision-based autonomous driving systems: Review and challenges,” 2021.
[3] P. Sun, H. Kretzschmar, X. Dotwilla, A. Chourad, V. Patnaik, P. Tsui, J. Gao, Y. Zhou, Y. Chai, B. Caine, V. Vasudevan, W. Han, J. Ngiam, H. Zhao, A. Timofeev, S. Ettinger, M. Krivokon, A. Gao, A. Joshi, Y. Zhang, J. Shilens, Z. Chen, and D. Anguelov. “Scalability in perception for autonomous driving: Waymo open dataset,” in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), June 2020.
[4] A. Tampuu, T. Matisen, M. Semikin, D. Fishman, and N. Muhammad, “A survey of end-to-end driving: Architectures and training methods,” IEEE Transactions on Neural Networks and Learning Systems, 2020.
[5] S. Griogrescu, B. Trasnea, T. Cocias, and G. Macesanu, “A survey of deep learning techniques for autonomous driving,” Journal of Field Robotics, vol. 37, no. 3, pp. 362–386, 2020.
[6] M. Bojarski, D. D. Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, X. Zhang, J. Zhao, and K. Zieba, “End to end learning for self-driving cars,” 2016.
[7] U. Muller, J. Ben, E. Cosatto, B. Flepp, and Y. L. Cun, “Off-road obstacle avoidance through end-to-end learning,” in Advances in neural information processing systems. Citeseer, 2006, pp. 739–746.
[8] F. Codevilla, M. Müller, A. López, V. Koltun, and A. Dosovitskiy, “End-to-end driving via conditional imitation learning,” in 2018 IEEE International Conference on Robotics and Automation (ICRA). IEEE, 2018, pp. 4693–4700.
[9] F. Codevilla, E. Santana, A. M. Lopez, and A. Gaidon, “Exploring the limitations of behavior cloning for autonomous driving,” in Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), October 2019.
[10] D. A. Pomerleau, “Alvinn: An autonomous land vehicle in a neural network,” CARNEGIE-MELLON UNIV PITTSBURGH PA ARTIFICIAL INTELLIGENCE AND PSYCHOLOGY …, Tech. Rep., 1989.
[11] S. Hecker, D. Dai, and L. Van Gool, “End-to-end learning of driving models with surround-view cameras and route planners,” in Proceedings of the European Conference on Computer Vision (ECCV), September 2018.
[12] S. Hecker, D. Dai, A. Liniger, M. Hahner, and L. Van Gool, “Learning accurate and human-like driving using semantic maps and attention,” in 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, 2020, pp. 2346–2353.
[13] J. Hawke, R. Shen, C. Gurau, S. Sharma, D. Reda, N. Nikolov, P. Mazur, S. Micklewhaite, N. Griffiths, A. Shah, and A. Kindall, “Urban driving with conditional imitation learning,” in 2020 IEEE International Conference on Robotics and Automation (ICRA), 2020, pp. 251–257.
[14] Y. Xiao, F. Codevilla, A. Gurram, O. Ufialiooglu, and A. M. Lopez, “Multimodal end-to-end autonomous driving,” IEEE Transactions on Intelligent Transportation Systems, pp. 1–11, 2020.
[15] Z. Huang, C. Lv, Y. Xing, and J. Wu, “Multi-modal sensor fusion-based deep neural network for end-to-end autonomous driving with scene understanding,” IEEE Sensors Journal, vol. 21, no. 10, pp. 11 781–11 790, 2021.
[16] A. Sauer, N. Savinov, and A. Geiger, “Conditional affordance learning for driving in urban environments,” in Proceedings of The 2nd Conference on Robot Learning, ser. Proceedings of Machine Learning Research, A. Billard, A. Dragan, J. Peters, and J. Morimoto, Eds., vol. 87. PMLR, 29–31 Oct 2018, pp. 237–252. [Online]. Available: http://proceedings.mlr.press/v87/sauer18a.html.
[17] J. Kim, T. Misu, Y.-T. Chen, A. Tawari, and J. Canny, “Grounding human-to-vehicle advice for self-driving vehicles,” in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), June 2019.
[18] D. Chen, B. Zhou, V. Koltun, and P. Krähenbühl, “Learning by cheating,” in Proceedings of the Conference on Robot Learning, ser. Proceedings of Machine Learning Research, L. P. Kaelbling, D. Kragic, and K. Sugiura, Eds., vol. 100. PMLR, 30 Oct–01 Nov 2020, pp. 66–75. [Online]. Available: http://proceedings.mlr.press/v100/chen20a.html

[19] P. Cai, H. Wang, Y. Sun, and M. Liu, “Learning scalable self-driving policies for generic traffic scenarios,” arXiv e-prints, pp. arXiv–2011, 2020.

[20] A. Prakash, A. Behl, E. Ohn-Bar, K. Chitta, and A. Geiger, “Exploring data aggregation in policy learning for vision-based urban autonomous driving,” in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), June 2020.

[21] M. Bansal, A. Krizhevsky, and A. Ogale, “Chauffeurnet: Learning to drive by imitating the best and synthesizing the worst,” 2018.

[22] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhudinov, R. Zemel, and Y. Bengio, “Show, attend and tell: Neural image caption generation with visual attention,” in International conference on machine learning. PMLR, 2015, pp. 2048–2057.