Novel Convolutional Transformer for Autonomous Driving

John Feng
j4480481@gmail.com

Abstract. A crucial component of an autonomous driving vehicle (ADV) is the machine learning, which is able to drive towards a desired destination. Nowadays, there are different paradigms addressing the development of ADV. On the one hand, the modular pipelines, which divide the driving task into sub-tasks such as perception and maneuver planning and control. On the other hand, the end-to-end driving approaches that try to learn a direct mapping from input raw sensor data to vehicle control signals. The later are relatively less studied, but are gaining popularity since they are simple and save computational cost. In this project, we focus on the end-to-end autonomous driving. How should representations from complementary modality be integrated for end-to-end autonomous driving? Recent multi-modal methods have shown that complementing RGB images with depth and semantics has the potential to improve driving performance. However, those methods can't fuse the multi-modal input well and may negatively affect driving performance. Therefore, in this project, we propose the multi-modal transformer to integrate the sensor information. We also apply the auto-regressive method for the temporal reasoning to predict the waypoint. Finally, in order to address the semantic gap between the waypoint values and senor features, we will tokenize the waypoint values and equip the model with the ability to learn the representation of them. We experimentally validate the efficacy of our approach in urban settings involving complex scenarios using the CARLA urban driving simulator. Our code is available in the attached files.

Keywords: autonomous driving, end-to-end, efficient inference, differentiable programming.

1 Introduction

The solutions to autonomous driving can be divided into two streams: Traditional Stack (Figure 1(a)) and end-to-end sensorimotor(Figure 1(b)). The traditional solution divides the overall driving task into multiple subtasks, including perception, prediction, planning, and control. Thanks to the rapid development of machine learning, the end-to-end solution can omit all the subtasks of the modular solution by building a mapping from high dimensional raw inputs of sensors directly to vehicle control command outputs [2]. In this way, the end-to-end solution can significantly reduce the complexity of the autonomous driving

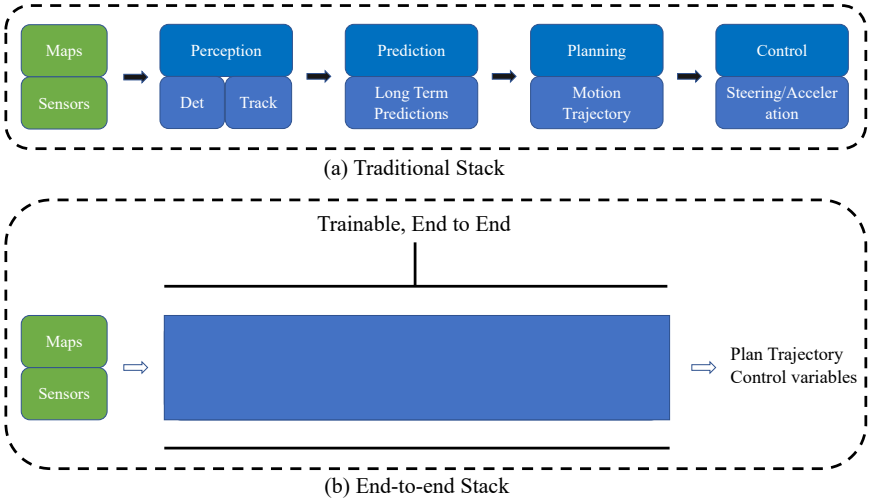

Fig. 1. Illustration of Traditional Stack and End-to-end Stack in autonomous driving. The traditional stack divides the overall driving task into multiple subtasks, including perception, prediction, planning, and control.

system while realizing the feasibility and effectiveness of the task objective. Recent end-to-end autonomous driving arises with the prosperity of deep learning techniques. It starts with a project from NVIDIA [2], in which the researchers used a convolutional neural network (CNN) that only takes as input the monocular image of the frontal road to steer a commercial vehicle. The end-to-end driving system has shown its ability in various driving conditions, including highways and rural roads. Inspired by this work, many works adopt the end-to-end approach to tackle autonomous driving problems, from lane-keeping [6] to driving in rural roads [15] and complexed urban areas [11].

End-to-end autonomous driving utilizes a deep neural network, which takes raw data from sensors (e.g., images from cameras, point cloud from Lidar) as inputs and outputs control commands (e.g. throttle, brake, and steering angle) or planing (e.g. trajectory), to drive a vehicle. Although it does not explicitly divide the driving task into several sub-modules, the network itself can be divided into two coupled parts in terms of functionalities, i.e. the environmental perception part and the driving policy part. The perception part is usually a CNN, which takes an image as the input and generates a low dimensional feature representation. The latent feature representation is subsequently connected to the driving policy module, which is typically a fully connected network, generating control commands as output. The vehicle control commands are used as the only supervision signals in a large amount of literature. These control signals are capable of supervising the network to learn driving commands while finding features related to driving tasks [33], such as lane markers and obsta-

cles [33]. However, such supervisions may not be strong enough to yield a good latent representation of the driving scene, and thus result in overfitting and lack of generalization capability and deteriorate the distributional shift problem [3]. Therefore, learning a good latent representation of the driving scene is of great importance for further improving the performance of end-to-end autonomous driving.

In this project, we focus on the end-to-end autonomous driving. How should representations from complementary sensors be integrated for end-to-end autonomous driving? Recent multi-modal methods have shown that complementing RGB images with depth and semantics has the potential to improve driving performance. However, those methods can't fuse the multi-modal input well and may negatively affect driving performance. Therefore, in this project, we propose the multi-modal transformer to integrate the sensor information. We also apply the auto-regressive method for the temporal reasoning to predict the waypoint. Finally, in order to address the semantic gap between the waypoint values and senor features, we will tokenize the waypoint values and equip the model with the ability to learn the representation of them.

To summarise, this work makes the following contributions,

1. We present a differentiable self-adaptive convolution (DSAC), which not only has powerful representation ability but also maintains model complexity. And DSAC can be used to replace the multi-model methods' traditional convolution in end-to-end autonomous driving.
2. We propose Multi-modal transformer to integrate the multi-modal input information, Auto-regressive method for the temporal reasoning to predict the waypoint and tokenized representation learning for waypoint to enhance the representation ability.
3. Our model obtain better performance than SOTAs on the overall driving score, especially on Collisions with pedestrians, Collisions with vehicles, Collisions with layout, Red lights infractions.

2 Related Work

2.1 End-to-End autonomous driving

Dosovitskiy et al. [9] introduced the CARLA driving simulator and demonstrated that a baseline end-to-end IL method with single camera input can achieve a performance comparable to a modular pipeline. After that, CIL [7] and CILRS [8] addressed directional multimodality in AD by using branched action heads where the branch is selected by a high-level directional command. While the aforementioned methods are trained via behavior cloning, DA-RB [24] applied DAGGER [26] with critical state sampling to CILRS. Most recently, LSD [23] increased the model capacity of CILRS by learning a mixture of experts and refining the mixture coefficients using evolutionary optimization. LBC proposed the mimicking methods aimed at training image-input networks with the supervision of a privileged model or squeezed model, which presents neural

attention fields to enable the reasoning for end-to-end driving models. Imitation learning(IL) approaches lack interpretability and their performance is limited by their handcrafted expert autopilot. [36] propose a novel fusion layer to efficiently extract features from vectorized High-Definition (HD) maps and utilize them in the end-to-end driving tasks. Here, we use DA-RB as the baseline IL agent to be supervised by Roach.

2.2 Dynamic Mechanism

With the prevalence of data dependency mechanism [1,14,28] , which emphasizes to extract more customized feature [22]. Studies about dynamic mechanism have promoted many tasks to new state-of-the-art. Benefiting from data dependency mechanism, networks can flexibly adjust themselves, including the structure and parameters, to fit the fickle information automatically and improve representation ability of neural networks. Some methods [4,31] indicate that different regions in the spatial dimension are not equally important in representation learning. For instance, activation in important regions needs to be amplified so that it can play dominant role in the forward propagation. SKNet [17] designs an efficacious module to channel-wisely select suitable receptive fields on the basis of channel attention and achieves better performance. It dynamically restructures the networks for the sake of different receptive fields in dilated convolutions [34,35]. In semantic segmentation, [37] imposes a pixel-group attention to remedy the deficiency of spatial information in SENet and [13] builds a link between each pixel and its surrounding pixels to capture important information. Attention mechanism is designed to dynamically calibrate the information flow in the forward propagation by learnable method.

From the aspect of dynamic weights, to handle object deformations, Deformable Kernels [10] directly resamples the original kernel space to adapt the effective receptive field (ERF) while leaving the receptive field untouched. Local Relation Networks [12] adaptively determines aggregation weights for spatial dimension based on the compositional relationship of local pixel pairs. Nonlocal [29] operation computes the response at each position as weighted sum of the features at all positions, which can make it to capture long-range dependencies.

2.3 Transformer model in autonomous driving

[16] utilize attention to capture temporal and spatial dependencies between actors by incorporating a transformer module into a recurrent neural network. [5] introduced a recurrent attention mechanism over a learned semantic map for predicting vehicle controls. However, it missed the multi-modal input features. [30] is a transformer work that learns an attention mask over features extracted from a 2D CNN, operating on LiDAR BEV projections and HD maps, to focus on dynamic agents for safe motion planning. TransFuser [25] exploits several transformer [28] modules for the fusion of intermediate features of front view and LiDAR. However, it failed to incorporate temporal reasoning. [21] proposes

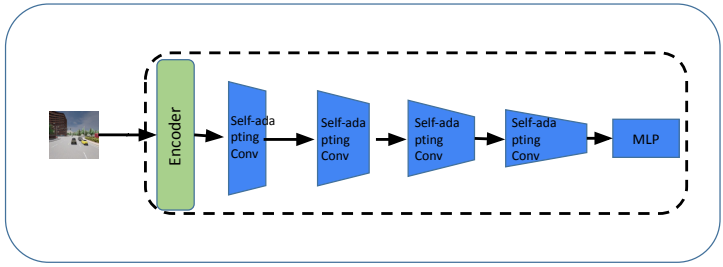

Fig. 2. Image feature extractor of our model. It consists of several self-adapting convolution modules.

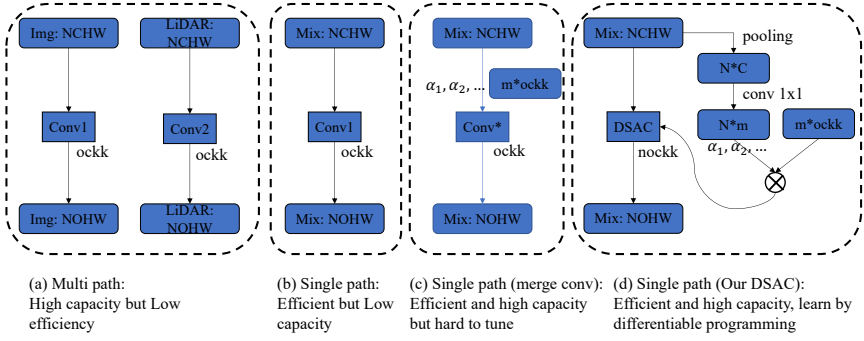

Fig. 3. Comparison of different convolutions.

Scene-Rep Transformer to improve the RL decision-making capabilities with better scene representation encoding and sequential predictive latent distillation. In [32], the encoded features are shared by the trajectory and multi-step control branch. Outputs from two branches are combined according to the situation based fusion scheme to generate the ultimate control actions.

3 Approach

In this section, we first describe the image feature extractor (Figure 2) and our novel self-adapting convolution module (DSAC in (Figure 3(d))). Then we will present how we construct the auto-regressive transformer with multi-modal input and the tokenized representation in (Figure 4).

3.1 Differentiable Self-adaptive Convolution (DSAC).

In a regular convolutional layer, the same convolutional kernel is used for all input examples (Figure 3(a,b,c)). In a DSAC layer, the convolutional kernel

is computed as a function of the input example (Figure 3(d)). Specifically, we parameterize the convolutional kernels in DSAC by:

$$\text{Output } (x) = \sigma\left((\alpha_1 \cdot W_1 + \ldots + \alpha_m \cdot W_m) * x\right) \tag{1}$$

where each $\alpha_i = r_i(x)$ is an example-dependent scalar weight computed using a routing function with learned parameters, m is the number of experts, and σ is an activation function. When we adapt a convolutional layer to use DSAC, each kernel W_i has the same dimensions as the kernel in the original convolution.

We typically increase the capacity of a regular convolutional layer by increasing the kernel height/width or number of input/output channels. However, each additional parameter in a convolution requires additional multiply-adds proportional to the number of pixels in the input feature map, which can be large. In a DSAC layer, we compute a convolutional kernel for each example as a linear combination of n experts before applying the convolution. Crucially, each convolutional kernel only needs to be computed once but is applied at many different positions in the input image. This means that by increasing n, we can increase the capacity of the network with only a small increase in inference cost; each additional parameter requires only 1 additional multiply-add.

A DSAC layer is mathematically equivalent to a more expensive linear mixture of experts formulation, where each expert corresponds to a static convolution (Figure 3):

$$\sigma\left((\alpha_1 \cdot W_1 + \ldots + \alpha_m \cdot W_m) * x\right) = \sigma\left(\alpha_1 \cdot (W_1 * x) + \ldots + \alpha_m \cdot (W_m * x)\right) \tag{2}$$

Thus, DSAC[19,20,18] has the same capacity as a linear mixture of experts formulation with n experts, but is computationally efficient since it requires computing only one expensive convolution. This formulation gives insight into the properties of DSAC and relates it to prior work on conditional computation and mixture of experts. The per-example routing function is crucial to DSAC performance: if the learned routing function is constant for all examples, a DSAC layer has the same capacity as a static convolutional layer.

We wish to design a per-example routing function that is computationally efficient, able to meaningfully differentiate between input examples, and is easily interpretable. We compute the example dependent routing weights $\alpha_i = r_i(x)$ from the layer input in three steps: global average pooling, fully-connected layer, Sigmoid activation.

$$r(x) = \text{Sigmoid}(\text{ GlobalAveragePool } (x)R), \tag{3}$$

where R is a matrix of learned routing weights mapping the pooled inputs to n expert weights. A normal convolution operation operates only over local receptive fields, so our routing function allows adaptation of local operations using global context.

3.2 Multi-modality Encoder.

To make use of the information aggregated in both image feature and Lidar feature, we follow it with a second operation which aims to fully capture channel-

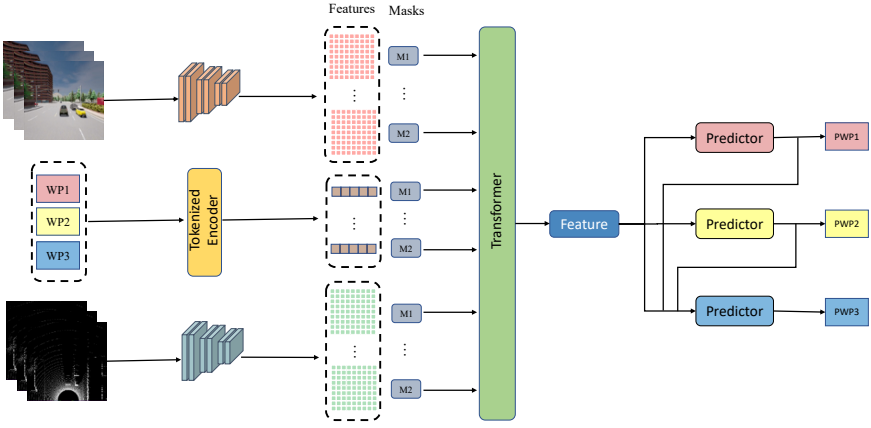

Fig. 4. Overview of our methods. We input the RGB image, LiDAR data and the previous way point feature to the temporal reasoning algorithm, and then auto-regressively predict the next way point. In order to address the semantic gap between the waypoint values and senor features, we tokenize the waypoint values for better representation. WP means way point and PWP means predicted way point. As for the mechanism of the transformer in the green block, we describes more in Figure 5.

wise dependencies. To fulfil this objective, the function must meet two criteria: first, it must be flexible (in particular, it must be capable of learning a nonlinear interaction between channels) and second, it must learn a non-mutually-exclusive relationship since we would like to ensure that multiple channels are allowed to be emphasised (rather than enforcing a one-hot activation). To meet these criteria, we opt to employ a simple gating mechanism with a sigmoid activation:

$$X_{il} = \text{Sigmoid}\left(W_2\delta\left(W_1 cat(X_i, X_l)\right)\right) \cdot cat(X_i, X_l), \tag{4}$$

where δ refers to the ReLU function. W_1, W_2 are parameters of two fully-connected (FC) layers. *cat* means concat and X_i denote image feature and X_l denotes Lidar feature. X_{fusion} is the fusion feature.

As for the plan trajectory input, we will use the tokenized representation X_{wp} of the ground truth waypoint for training, and predicted one for testing. We can utilze the two PID controllers for lateral and longitudinal control to obtain steer, throttle and brake values from the predicted waypoints. This is motivated by the fact that additional controllers such as PID controllers are usually needed as a subsequent step to convert the planned trajectory into control signals. However, turning the trajectory into control actions so that the vehicle could follow the planned trajectory is not trivial.

$$X_{fusion} = \text{Cross-Transformer}(X_{il}, X_{wp}) \tag{5}$$

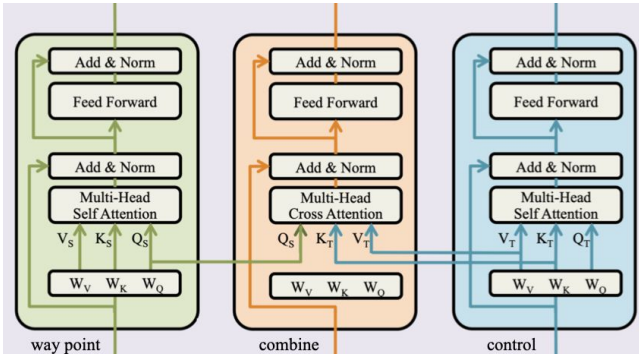

Fig. 5. Illustration of the cross-transformer in Figure 4, which has three branches.

After getting the representations from the RGB image, LiDAR data and the previous way point, we will auto-regressively predict the next way point. To achieve that, we aggregate the three features by using cross transformer encoder Figure 5, which has three branches. The first and third branch is just like normal transformer layer with the multi-head attention. In the first branch, given the way point representation as the query, key and value, we feed them into the multi-head attention. Then after several linear layers and layer norm, the new contextual way point representation is generated. The third branch with X_{il} features is similar. As for the mid-branch, we regard the way point representation as the query and use the X_{il} features as the key and value. Finally, we get X_{fusion} by concatenating all the output.

To generate the final output way point, we follow the waypoint prediction work proposed by [25], we pass the 512-dimensional feature vector X_{fusion} through an MLP (comprising 2 hidden layers with 256 and 128 units) to reduce its dimensionality to 64 for computational efficiency before passing it to the auto-regressive waypoint network implemented using GRUs. We initialize the hidden state of the GRU with the 64-dimensional feature vector. The update gate of the GRU controls the flow of information encoded in the hidden state to the output and the next time-step. It also takes in the current position and the goal location as input, which allows the network to focus on the relevant context in the hidden state for predicting the next waypoint. As for the loss function, following the setting of [25], we train the network using an $L1$ loss between the predicted waypoints and the ground truth way points (from the expert), registered to the current coordinate frame.

4 Experiment

4.1 Dataset

To prove the effectiveness of our method, we do experiments on way points prediction task. Following the setting of [25], we consider the task of navigation

Table 1. Ablation study on long routes way points prediction with different image feature extractors. PC: point cloud, DS: Avg. driving score, RC: Avg. route completion, IP: Avg. infraction penalty, CP: Collisions with pedestrians, CV: Collisions with vehicles, CL: Collisions with layout, RLI: Red lights infractions, SSI: Stop sign infractions, trad. conv: traditional convolution, non-diff: non-differentiable.

Model	DS↑	RC↑	IP↓	CP↓	CV↓	CL↓	RLI↓	SSI↓
Image-single-model	5.9	22.8	0.47	0.00	0.32	0.63	0.18	0.05
Image-single-model(DSAC, Ours)	12.9	**45.6**	**0.41**	0.00	0.08	0.85	0.30	0.01
Image+PC-two-model	8.3	38.8	0.50	0.00	0.09	0.93	0.18	0.01
Image+PC-single-model (trad. conv)	10.2	33.3	0.47	0.00	0.26	1.32	0.12	0.04
Image+PC-single-model (non-diff)	10.8	12.4	0.83	0.00	0.06	0.56	0.18	0.00
Image+PC-single-model (DSAC, Ours)	**17.1**	24.8	0.84	0.00	**0.05**	**0.00**	**0.09**	**0.00**

along a set of predefined routes in different areas, such as motorways, urban regions, and residential districts. A sequence of sparse goal locations in GPS coordinates provided by a global planner and the related discrete navigational commands, such as "follow lane", "turn left/right", and "change lane", constitute the routes. Only the sparse GPS locations are used in our method. Each route is constituted of several scenarios that are initialized at predefined locations and test the agent's ability to handle various adversarial situations, such as obstacle avoidance, unprotected turns at intersections, vehicles running red lights, and pedestrians emerging from occluded areas crossing the road at random locations. The agent needs to complete the route within a certain amount of time, while following traffic restrictions and dealing with large numbers of dynamic agents. For dataset, we use the CARLA [9] simulator for training, testing and visualization, specifically CARLA 0.9.10 which consists of 8 publicly available towns. We use 7 towns for training and hold out Town05 for evaluation as in [25].

4.2 Experiment Details

Metrics. We reported eight metrics. PC: point cloud, DS: Avg. driving score, RC: Avg. route completion, IP: Avg. infraction penalty, CP: Collisions with pedestrians, CV: Collisions with vehicles, CL: Collisions with layout, RLI: Red lights infractions, SSI: Stop sign infractions, trad. conv: traditional convolution, non-diff: non-differentiable.

Implementation Details. We utilize python as the programming language and pytorch as the framework. We use 2 sensor modalities, the front camera RGB image and LiDAR point cloud converted to BEV representation, i.e., $S = 2$. The RGB image and LiDAR BEV encoded using a ResNet-18 [27] which is trained from scratch. For more details about the code and environment, please refer to our repo.

Table 2. Performance comparison between baseline methods on long routes way points prediction. PC: point cloud, DS: Avg. driving score, RC: Avg. route completion, IP: Avg. infraction penalty, CP: Collisions with pedestrians, CV: Collisions with vehicles, CL: Collisions with layout, RLI: Red lights infractions, SSI: Stop sign infractions, trad. conv: traditional convolution, non-diff: non-differentiable.

Model	DS↑	RC↑	IP↓	CP↓	CV↓	CL↓	RLI↓	SSI↓
GRIAD	36.8	61.9	0.60	0.00	2.77	0.41	0.48	0.00
LAV	61.9	94.5	0.64	0.04	0.70	0.02	0.17	0.00
TransFuser	61.2	86.7	0.71	0.04	0.81	0.02	0.05	0.00
TCP	73.3	96.1	0.77	0.07	0.04	0.02	0.04	0.05
SAT (Ours)	**74.6**	95.3	0.79	**0.03**	**0.01**	**0.02**	**0.01**	0.06

Baselines. *TransFuser* [25] exploits several transformer [28] modules for the fusion of intermediate features of front view and LiDAR. [32] introduce *TCP*, the encoded features are shared by the trajectory and multi-step control branch. Outputs from two branches are combined according to the situation based fusion scheme to generate the ultimate control actions. *GRIAD* is first trained using expert supervision to predict future waypoints followed by an image-based student model which is trained using supervision from the teacher. *LAV* is a conditional imitation learning method in which the agent learns to predict vehicle controls from a single front camera image while being conditioned on the navigational command.

Results. As shown in Table 1, we have verified our method in single model paradigm, our experiment show that single model based method applied with our DSAC can obtain a significant improvement. Furthermore, we compared with the multi-model method, we have achieved big improvement even in a single model paradigm, which means we only use about 53% computational cost of multi-model method and we can achieve much better performance. Thus, our model can achieve better performance in both single and multi model methods. Moreover, our differentiable method outperform the traditional convolution and non-differentiable methods. Compare between Table 1 and Table 2, the performance in Table 1 is much lower than the results in Table 2, which is because we only use the front camera RGB image and LiDAR point cloud as the input and early fusion attention module to combine the features instead of the cross-modal transformer encoder. which demonstrate its effectiveness. As for the comparison experiment results in Table 2, we obtain better performance than SOTAs on the overall driving score, especially on Collisions with pedestrians, Collisions with vehicles, Collisions with layout, Red lights infractions.

5 Conclusion

In this project, we propose differentiable self-adaptive convolution (DSAC), which can efficiently increase the capacity of a convolutional layer. DSAC allows us to increase model capacity and performance while maintaining efficient

inference. The proposed DSAC can entirely become substitute of standard convolution in any existing networks. Furthermore, We also propose the tokenized representation learning for waypoint to enhance the representation ability, multi-modal transformer to integrate the multi-modal sensor information and auto-regressively predict the waypoint by the temporal reasoning. Comprehensive experiments on different paradigms have shown the effectiveness our model on the overall driving score, while maintaining efficient inference. In the future, we plan to test our model on different datasets. In addition, we will combine other objective functions, including objection detection and semantic segmentation, and make it a multi-task learning strategy to improve the performance.

References

1. Allport, A.: Visual attention. The MIT Press (1989)
2. Bojarski, M., Del Testa, D., Dworakowski, D., Firner, B., Flepp, B., Goyal, P., Jackel, L.D., Monfort, M., Muller, U., Zhang, J., et al.: End to end learning for self-driving cars. arXiv preprint arXiv:1604.07316 (2016)
3. Bojarski, M., Yeres, P., Choromanska, A., Choromanski, K., Firner, B., Jackel, L., Muller, U.: Explaining how a deep neural network trained with end-to-end learning steers a car. arXiv preprint arXiv:1704.07911 (2017)
4. Chen, L., Zhang, H., Xiao, J., Nie, L., Shao, J., Liu, W., Chua, T.S.: Sca-cnn: Spatial and channel-wise attention in convolutional networks for image captioning. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 5659–5667 (2017)
5. Chen, S., Zhang, S., Shang, J., Chen, B., Zheng, N.: Brain-inspired cognitive model with attention for self-driving cars. IEEE Transactions on Cognitive and Developmental Systems **11**(1), 13–25 (2017)
6. Chen, Z., Huang, X.: End-to-end learning for lane keeping of self-driving cars. In: 2017 IEEE Intelligent Vehicles Symposium (IV). pp. 1856–1860. IEEE (2017)
7. Codevilla, F., Müller, M., López, A., Koltun, V., Dosovitskiy, A.: End-to-end driving via conditional imitation learning. In: 2018 IEEE International Conference on Robotics and Automation (ICRA). pp. 4693–4700. IEEE (2018)
8. Codevilla, F., Santana, E., López, A.M., Gaidon, A.: Exploring the limitations of behavior cloning for autonomous driving. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 9329–9338 (2019)
9. Dosovitskiy, A., Ros, G., Codevilla, F., Lopez, A., Koltun, V.: Carla: An open urban driving simulator. In: Conference on robot learning. pp. 1–16. PMLR (2017)
10. Gao, H., Zhu, X., Lin, S., Dai, J.: Deformable kernels: Adapting effective receptive fields for object deformation. arXiv preprint arXiv:1910.02940 (2019)
11. Hawke, J., Shen, R., Gurau, C., Sharma, S., Reda, D., Nikolov, N., Mazur, P., Micklethwaite, S., Griffiths, N., Shah, A., et al.: Urban driving with conditional imitation learning. In: 2020 IEEE International Conference on Robotics and Automation (ICRA). pp. 251–257. IEEE (2020)
12. Hu, H., Zhang, Z., Xie, Z., Lin, S.: Local relation networks for image recognition. In: IEEE International Conference on Computer Vision (ICCV). pp. 3464–3473 (2019)
13. Huang, Z., Wang, X., Huang, L., Huang, C., Wei, Y., Liu, W.: Ccnet: Criss-cross attention for semantic segmentation. In: IEEE International Conference on Computer Vision (ICCV). pp. 603–612 (2019)

14. Jaderberg, M., Simonyan, K., Zisserman, A., et al.: Spatial transformer networks. In: Advances in Neural Information Processing Systems (NeurIPS). pp. 2017–2025 (2015)
15. Kendall, A., Hawke, J., Janz, D., Mazur, P., Reda, D., Allen, J.M., Lam, V.D., Bewley, A., Shah, A.: Learning to drive in a day. In: 2019 International Conference on Robotics and Automation (ICRA). pp. 8248–8254. IEEE (2019)
16. Li, L.L., Yang, B., Liang, M., Zeng, W., Ren, M., Segal, S., Urtasun, R.: End-to-end contextual perception and prediction with interaction transformer. In: 2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). pp. 5784–5791. IEEE (2020)
17. Li, X., Wang, W., Hu, X., Yang, J.: Selective kernel networks. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 510–519 (2019)
18. Liu, F., Lin, K., Li, L., Wang, J., Yacoob, Y., Wang, L.: Aligning large multi-modal model with robust instruction tuning. arXiv preprint arXiv:2306.14565 (2023)
19. Liu, F., Wang, Y., Wang, T., Ordonez, V.: Visual news: Benchmark and challenges in news image captioning. arXiv preprint arXiv:2010.03743 (2020)
20. Liu, F., Wang, Y., Wang, T., Ordonez, V.: Visualnews: Benchmark and challenges in entity-aware image captioning. arXiv preprint arXiv:2010.03743 (2020)
21. Liu, H., Huang, Z., Mo, X., Lv, C.: Augmenting reinforcement learning with transformer-based scene representation learning for decision-making of autonomous driving. arXiv preprint arXiv:2208.12263 (2022)
22. Mahajan, D., Girshick, R., Ramanathan, V., He, K., Paluri, M., Li, Y., Bharambe, A., van der Maaten, L.: Exploring the limits of weakly supervised pretraining. In: European Conference on Computer Vision (ECCV). pp. 181–196 (2018)
23. Ohn-Bar, E., Prakash, A., Behl, A., Chitta, K., Geiger, A.: Learning situational driving. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 11296–11305 (2020)
24. Prakash, A., Behl, A., Ohn-Bar, E., Chitta, K., Geiger, A.: 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. pp. 11763–11773 (2020)
25. Prakash, A., Chitta, K., Geiger, A.: Multi-modal fusion transformer for end-to-end autonomous driving. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 7077–7087 (2021)
26. Ross, S., Gordon, G., Bagnell, D.: A reduction of imitation learning and structured prediction to no-regret online learning. In: Proceedings of the fourteenth international conference on artificial intelligence and statistics. pp. 627–635. JMLR Workshop and Conference Proceedings (2011)
27. Targ, S., Almeida, D., Lyman, K.: Resnet in resnet: Generalizing residual architectures. arXiv preprint arXiv:1603.08029 (2016)
28. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. In: Advances in Neural Information Processing Systems (NeurIPS). pp. 5998–6008 (2017)
29. Wang, X., Girshick, R., Gupta, A., He, K.: Non-local neural networks. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 7794–7803 (2018)
30. Wei, B., Ren, M., Zeng, W., Liang, M., Yang, B., Urtasun, R.: Perceive, attend, and drive: Learning spatial attention for safe self-driving. In: 2021 IEEE International Conference on Robotics and Automation (ICRA). pp. 4875–4881. IEEE (2021)
31. Woo, S., Park, J., Lee, J.Y., So Kweon, I.: Cbam: Convolutional block attention module. In: European Conference on Computer Vision (ECCV). pp. 3–19 (2018)

32. Wu, P., Jia, X., Chen, L., Yan, J., Li, H., Qiao, Y.: Trajectory-guided control prediction for end-to-end autonomous driving: A simple yet strong baseline. arXiv preprint arXiv:2206.08129 (2022)
33. Yang, S., Wang, W., Liu, C., Deng, W., Hedrick, J.K.: Feature analysis and selection for training an end-to-end autonomous vehicle controller using deep learning approach. In: 2017 IEEE Intelligent Vehicles Symposium (IV). pp. 1033–1038. IEEE (2017)
34. Yu, F., Koltun, V.: Multi-scale context aggregation by dilated convolutions. arXiv preprint arXiv:1511.07122 (2015)
35. Yu, F., Koltun, V., Funkhouser, T.: Dilated residual networks. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 472–480 (2017)
36. Zhang, Q., Tang, M., Geng, R., Chen, F., Xin, R., Wang, L.: Mmfn: Multi-modal-fusion-net for end-to-end driving. arXiv preprint arXiv:2207.00186 (2022)
37. Zhong, Z., Lin, Z.Q., Bidart, R., Hu, X., Daya, I.B., Li, J., Wong, A.: Squeeze-and-attention networks for semantic segmentation. arXiv preprint arXiv:1909.03402 (2019)