Supplementary Material for *DriveAdapter: Breaking the Coupling Barrier of Perception and Planning in End-to-End Autonomous Driving*

Xiaosong Jia[1,2], Yulu Gao[2,3], Li Chen[2], Junchi Yan[1,2†], Patrick Langechuan Liu[4], Hongyang Li[2,1†]

[1] MoE Key Lab of Artificial Intelligence, Shanghai Jiao Tong University
[2] OpenDriveLab, Shanghai AI Lab [3] Beihang University [4] Anker Innovations
[†]Correspondence authors

https://github.com/OpenDriveLab/DriveAdapter

1. Case Study of Causal Confusion

As discussed in the introduction of the paper, behavior cloning based methods suffer from the *inertia issue* [3]. As shown in Fig. 1, at the intersection, if the ego vehicle stops at the front of the line, other vehicles have to wait for its movement. However, since all surrounding vehicles do not move, the ego vehicle would not move since it learns the improper correlation. This is a typical example of *causal confusion*.

In TeacherAdapter, there is no such situation at all. The route mask in the BEV segmentation is generated by drawing instead of prediction and is fed into frozen the teacher model. Thus, the model naturally has the tendency to move. By comparing with the *Unfrozen Teacher Model* variant in Table 5 in the paper and *Action* variant in Table 6 in the paper , we could find that TeacherAdapter has much higher route completion (RC) score which demonstrates the superiority of utilizing the knowledge within the teacher model over behavior cloning. Also, choosing all learning targets of the student model except *Action* in Table 6 in the paper have high route completion since they all use part of the teacher model.

In summary, by adopting the frozen teacher model for decision-making, the causal confusion issue is avoided and the student model could focus on feature extraction and perception learning. However, there are still some failure cases under this paradigm and we give a thorough investigation of them in the supplemental materials.

2. Implementation Details

Since an end-to-end autonomous driving model is a large system, we provide details of our implementation in terms of data collection, model configuration, training hyperparameters, and data augmentation. We will make the code and model publicly available.

2.1. Data Collection

We use Roach [18] as the expert with a collision detector for emergency stop similar to [16]. We set the following sensors:

We set four cameras with field of view (FOV) $150°$: Front (x=1.5, y=0.0, z=2.5, yaw=$0°$), Left (x=0.0, y=-0.3, z=2.5,yaw=$-90°$), Right (x=0.0, y=0.3, z=2.5, yaw=$90°$), Back(x=-1.6, y=0.0, z=2.5, yaw=$180°$) where the aforementioned coordinate and angle are all in the ego coordinate system. The output of each camera is a 900x1600 RGB image. Since CARLA [6] simulates the Brown-Conrady distortion [4], we estimate the distortion parameter with the code of [15]. The estimated parameter for the distortion is (0.00888296, -0.00130899, 0.00012061, -0.00338673, 0.00028834) and we use the parameter to calibrate images before we feed them into the neural network. We also collected the depth and semantic segmentation label of images.

We set one Lidar with 64 channels, upper FOV $10°$, lower FOV $-20°$, and frequency 10Hz, following the official protocol. We set it at (x=0.0, y=0.0, z=2.5, yaw=$0°$).

We set an IMU to estimate the yaw angle, acceleration, and angular velocity of the ego vehicle. We set a GPS to estimate current world coordinate of the ego vehicle and a speedometer to estimate current speed of the ego vehicle.

Following the official setting, we also save the target point which might be hundreds meters away as well high-level commands (keep straight, turn left, turn right, etc) provided by the protocol.

To conduct feature alignment, we collect feature maps of Roach at different layers. To conduct action guidance, we store the final action and whether the learning-based model is overridden by the rules.

We convert all raw data into the ego coordinate system.

Figure 1: Visualization of the inertia issue where the behavior cloning-based agents often fail. The agent fails to proceed despite the green traffic light, blocking the other car behind.

2.2. Model Configuration

Our code is based on OpenMMLab [5] with Pytorch [12], where we use their official implementation of backbones and cooresponding ImageNet pretrained weights if applicable. We use ResNet50 [7] as the image backbone. We use the PAFPN [10] to obtain the multi-scale image features. As for the LSS [13] and depth module, we adopt the code from [9]. We use a U-Net [14]-like structure for the image semantic segmentation, similar to [1, 3]. We downsample all images to 450x800 to save GPU memory. For the Lidar model, we use the SECOND [17] implemented by mmdetection3D which consists of HardSimpleVFE, SparseEncoder, SECOND, and SECONDFPN. We use 2 frames as the input. The BEV grid of both modalities is 256x256 and the scale is (Front=36.8m, Back=-14.4m, Left=-25.6m, Right=25.6m). We conduct the BEV feautres of the two modalities and use a series of CNN to fuse them. Finally, we only keep the center 192x192 to conduct BEV segmentation which matches with the input of Roach. To capture the dynamics of surrounding agents so that we could segment past agents' position, in both Lidar (SECOND) and camera backbones (LSS), we stack 1 extra history frame, which means the input contains two frames [-1, 0]. Specifically, in SECOND, we simply stack all point clouds and add a additional channel to indicate the time-step. In LSS, we transformer the history BEV feature into the current ego coordinate system and concate it with current BEV feature and finally feed them into Conv layers.

As for the BEV segmentation, Roach's privileged inputs have 24 types: road mask, route mask, lane mask, vehicle mask (-3, -2, -1, 0), pedestrian mask (-3, -2, -1, 0), traffic light mask (-3, -2, -1, 0) where (-3, -2, -1, 0) denotes the history timestep in 2Hz. Note that lane and broken lane are represented by 1.0 and 0.5 respectively in the lane mask. Green, Yellow, and Red light are 0.3137, 0.6667, and 1.0 respectively in the traffic light mask. To formulate the task as BEV segmetation, we separate those mixed classes into independent classes and turn the prediction results back when feeding into the teacher model. Additionally, since the route information is given, it is unnecessary to predict the mask and thus we just need to draw the route mask. We adopt the Mask2former [2] for semantic segmentation with the BEV feature as inputs based on their official implementation. We use only one scale (192x192), 3 encoder layers and 6 decoder layers. .We treat each object type at each timestep as a class and set corresponding queries to conduct segmentation. In the 189K frames setting, we only use 4 towns (no overlap with Town05Long and some overlap with Longest6) to match with Roach, TCP, LAV, etc. While in the 2M frames settings*, we use all 8 towns (overlap with both Town05Long and Longest6). We observe more accurate BEV segmentation results and less red light infraction in 2M frames settings due to more data and seen towns.

As for the adapter module, for each 2D feature map, the adapter is one Resnet bottleneck with SE [8]. For each 1D feature map, the adapter is a two-layer MLP.

As for the *+TCP* setting, we add an MLP with the 1D feature map of the last layer as inputs and use the trajectory generated by the expert as labels.

The total number of parameters is 135M, the MACs is 1719G, and the inference GPU memory is around 5G.

2.3. Hyper-Parameters

We use AdamW [11] optimizer with the learning rate 1e-4, cosine learning rate decay, effective batch size 96, and weight decay 1e-7. We train the model for 60 epochs. For hidden dimensions, we use 256 at most places. For loss weights, we tune them to make sure that each loss is around 1 at the beginning of training. We apply gradient clip based on the L2 norm with the threshold of 35.

2.4. Data Augmentation

For data augmentation which we only apply on images, we use the random color transformation similar to [16] and random crop before we project image features to the BEV grid.

References

[1] Dian Chen and Philipp Krähenbühl. Learning from all vehicles. In *CVPR*, 2022. 2

[2] Bowen Cheng, Ishan Misra, Alexander G. Schwing, Alexander Kirillov, and Rohit Girdhar. Masked-attention mask transformer for universal image segmentation. In *CVPR*, pages 1290–1299, 2022. 2

[3] Kashyap Chitta, Aditya Prakash, Bernhard Jaeger, Zehao Yu, Katrin Renz, and Andreas Geiger. Transfuser: imitation with transformer-based sensor fusion for autonomous driving. *TPAMI*, 2022. 1, 2

[4] A. E. Conrady. Decentred Lens-Systems. *Monthly Notices of the Royal Astronomical Society*, 79(5):384–390, 03 1919. 1

[5] MMCV Contributors. MMCV: OpenMMLab computer vision foundation. https://github.com/open-mmlab/mmcv, 2018. 2

[6] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. CARLA: an open urban driving simulator. In *CoRL*, pages 1–16, 2017. 1

[7] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016. 2

[8] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *CVPR*, pages 7132–7141, 2018. 2

[9] Yinhao Li, Zheng Ge, Guanyi Yu, Jinrong Yang, Zengran Wang, Yukang Shi, Jianjian Sun, and Zeming Li. Bevdepth: acquisition of reliable depth for multi-view 3d object detection. *AAAI*, 2023. 2

[10] Shu Liu, Lu Qi, Haifang Qin, Jianping Shi, and Jiaya Jia. Path aggregation network for instance segmentation. In *CVPR*, pages 8759–8768, 2018. 2

[11] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *ICLR*, 2018. 2

[12] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: an imperative style, high-performance deep learning library. *NeurIPS*, 32, 2019. 2

[13] Jonah Philion and Sanja Fidler. Lift, splat, shoot: Encoding images from arbitrary camera rigs by implicitly unprojecting to 3d. In *ECCV*, pages 194–210. Springer, 2020. 2

[14] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: convolutional networks for biomedical image segmentation. In *MICCAI*, pages 234–241. Springer, 2015. 2

[15] Abanob Soliman, Fabien Bonardi, Désiré Sidibé, and Samia Bouchafa. IBISCape: a simulated benchmark for multi-modal SLAM systems evaluation in large-scale dynamic environments. *Journal of Intelligent & Robotic Systems*, 106(3):53, Oct 2022. 1

[16] Penghao Wu, Xiaosong Jia, Li Chen, Junchi Yan, Hongyang Li, and Yu Qiao. Trajectory-guided control prediction for end-to-end autonomous driving: a simple yet strong baseline. *NeurIPS*, 2022. 1, 2

[17] Yan Yan, Yuxing Mao, and Bo Li. Second: sparsely embedded convolutional detection. *Sensors*, 18(10):3337, 2018. 2

[18] Zhejun Zhang, Alexander Liniger, Dengxin Dai, Fisher Yu, and Luc Van Gool. End-to-end urban driving by imitating a reinforcement learning coach. In *ICCV*, 2021. 1