Supplementary Material for *Think Twice before Driving: Towards Scalable Decoders for End-to-End Autonomous Driving*

Xiaosong Jia[1,2], Penghao Wu[2,3], Li Chen[2], Jiangwei Xie[4], Conghui He[2], Junchi Yan[1,2†], Hongyang Li[2,1†]

[1] Shanghai Jiao Tong University [2] Shanghai AI Laboratory
[3] University of California at San Diego [4] SenseTime Research
[†]Correspondence authors

1. Implementation Details

Since an end-to-end autonomous driving model is a large system, we provide details of our implementation so that it is easier for the community to reproduce. We will make the code and model publicly available.

1.1. Data Collection

We use Roach [24] as the expert with a collision detector for emergency stop similar to [22]. 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 [7] simulates the Brown-Conrady distortion [5], we estimate the distortion parameter with the code of [21]. 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.

As for additional supervision signals, we save the value function, BEV feature maps of different resolution, the 1D feature as well as the control actions of Roach.

We convert all raw data into the ego coordinate system.

1.2. Models

Our code is based on OpenMMLab [6] with Pytorch [17], where we use their official implementation of backbones and cooresponding ImageNet pretrained weights if applicable. We use ResNet50 [8] as the image backbone. For the extra data settings, we use ConvNext-base [14]. We use the PAFPN [13] to obtain the multi-scale image features. As for the LSS [18] and depth module, we adopt the code from [12]. For the semantic segmentation module, we use a U-Net [19]-like structure. We downsample all images to 450x800 to save GPU memory. We use the BEV grid of 21x21 for low computational burden and set the scale as (Front=30.4m, Back=-8.0m, Left=-19.2m, Right=19.2m) which matches with the scale of Roach. We use 2 frames as the input while for the extra data settings we use 3 frames. For the Lidar model, we use the SECOND [23] implemented by mmdetection3D which consists of HardSimpleVFE, SparseEncoder, SECOND, and SECONDFPN. For the decoder, we have described details in the main text. For the extra data setting, we train 3 heads to further enlarge the capacity of the decoder.

1.3. Hyper-Parameters

We use AdamW [16] optimizer with the learning rate 1e-4, cosine learning rate decay, effective batch size 128, 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.

1.4. Other Details

For data augmentation which we only apply on images, we use the random color transformation similar to [22] and

Method	Encoder	Decoder	Modality	#Parameters	MACs (G)	GPU Memory
CILRS [4]	ResNet + Flatten	MLP	C1	23.4M	1.4	1507M
LBC [2]	ResNet + Flatten	MLP	C3	23.1M	5.4	1627M
Transfuser [3]	Fusion via Transformer	GRU	C3L1	165.8M	34.9	2755M
Roach [24]	ResNet + Flatten	MLP	C1	23.4M	17.1	2171M
LAV [1]	PointPaiting	Multi-layer GRUs	C4L1	27.5M	45.5	2493M
TCP [22]	ResNet + Flatten	GRU	C1	25.8M	17.1	2177M
MILE [9]	ResNet + Flatten	GRU	C1	54.1M	11.2	2485M
Interfuser [20]	Fusion via Transformer	Transformer + GRU	C3L1	82.8M	46.5	1823M
ThinkTwice	Geometric Fusion in BEV	Look-Predict-Refine	C4L1	120.2M	1170+45	4019M

Table 1. **Comparison of Computational Burden** MACs and #Parameters are calculated by the Python package *thop*. MACs and GPU memory is calculated under the inference mode of models. For ThinkTwice, the MACs is 1170G for the encoder (LSS module = 1157G) For *Modality*, C denotes the camera sensor and L denotes the Lidar sensor.

random crop before we project image features to the BEV grid. We stop the augmentation at the last ten epochs. The prediction time-horizon is 4. We apply gradient clip based on the L2 norm with the threshold of 35.

2. Discussion about Inference Computation

For the deployment of autonomous driving models, different from running on the cluster, they are usually running on edge devices with limited computational power and memory. Thus, it is important to discuss about the computation requirement and the memory footprint of different models during inference. Since few existing works give the computation related statistics in their papers, we run their official code to obtain an estimation. Note that all models are under their officially suggested environment, which means the version of packages such as Pytorch, Cuda, Carla, Numpy, OpenCV, etc may have some influence. All models are running under the same server with an RTX 3090 GPU. The estimation are in the Table 1. We could observe that ThinkTwice has large MACs and GPU memory usage. It is because ThinkTwice is the only model adopts geometric fusion in BEV - specifically, LSS [18] which requires 1153G MACs during the inference. We adopt it since the BEV representation inherently preserves the spatial relationships on the ground plane, making it preferable for joint perception-planning and sensor fusion. From Table 4 in the main text, we could observe that simply combining the BEV representation with TCP [22] could achieve59 DS and the performance could be further enhanced to 65 DS with the proposed coarse-to-fine decoder. Actually, learning BEV representation is a heated topic in both industry and academia. To reduce the heavy computational burden of BEV-based model, more efficient implementations have been proposed in BEVFusion [15], BEVDepth [12], BEVStereo [11], BEV-Pool-V2 [10], etc. Specifically designed edge devices and chips are also actively explored by the industry.

3. Visualization

We visualize different layers of predictions in the Fig. 1. We can observe that with the future conditioned coarse-to-fine refinement, the driving process is safer and smoother. We also provide a **video** in the supplemental materials with diverse scenarios about the driving process of ThinkTwice.

References

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

[2] Dian Chen, Brady Zhou, Vladlen Koltun, and Philipp Krähenbühl. Learning by cheating. In *Conference on Robot Learning*, pages 66–75. PMLR, 2020. 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. *Pattern Analysis and Machine Intelligence (PAMI)*, 2022. 2

[4] Felipe Codevilla, Eder Santana, Antonio M López, and Adrien Gaidon. Exploring the limitations of behavior cloning for autonomous driving. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9329–9338, 2019. 2

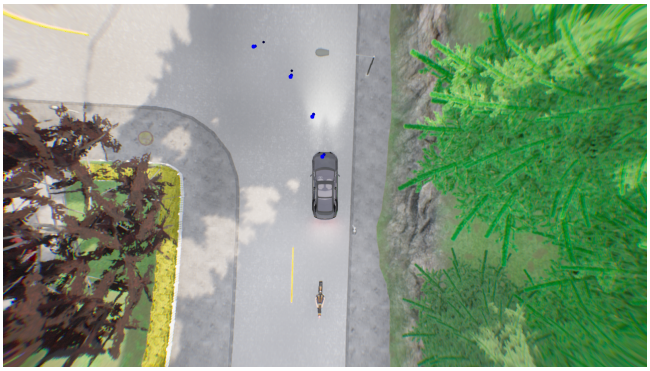
[5] A. E. Conrady. Decentred Lens-Systems. *Monthly Notices of the Royal Astronomical Society*, 79(5):384–390, 03 1919. 1

[6] MMCV Contributors. MMCV: OpenMMLab computer vision foundation. https://github.com/open-mmlab/mmcv, 2018. 1

[7] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. CARLA: An open urban driving simulator. In *Proceedings of the 1st Annual Conference on Robot Learning*, pages 1–16, 2017. 1

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 1

[9] Anthony Hu, Gianluca Corrado, Nicolas Griffiths, Zak Murez, Corina Gurau, Hudson Yeo, Alex Kendall, Roberto Cipolla, and Jamie Shotton. Model-based imitation learning for urban driving. *arXiv preprint arXiv:2210.07729*, 2022. 2



(a) **Quarter Turn**. It is a common failure case due to its sharp turn angle. Here, the refined trajectory has a more accurate turning route.

(b) **Emergency Stop**. For the jay-walker with the nearby vehicle's occlusion, the refined trajectory leads to a deacceleration compared to the original one.

(c) **Merge**. The refined trajectory leaves more advance for the merging, which leads to a safer and smoother driving.

(d) **Lane Changing.** The refined trajectory notices the jay-walker and leads to a emergency stop during the lane-changing process.

Figure 1. Visualization for the predictions from different layers of decoder. Larger and brighter dots are from deeper layers.

[10] Junjie Huang and Guan Huang. Bevpoolv2: A cutting-edge implementation of bevdet toward deployment. *arXiv preprint arXiv:2211.17111*, 2022. 2

[11] Yinhao Li, Han Bao, Zheng Ge, Jinrong Yang, Jianjian Sun, and Zeming Li. Bevstereo: Enhancing depth estimation in multi-view 3d object detection with dynamic temporal stereo. *arXiv preprint arXiv:2209.10248*, 2022. 2

[12] 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. *arXiv preprint arXiv:2206.10092*, 2022. 1, 2

[13] Shu Liu, Lu Qi, Haifang Qin, Jianping Shi, and Jiaya Jia. Path aggregation network for instance segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8759–8768, 2018. 1

[14] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. 1

[15] Zhijian Liu, Haotian Tang, Alexander Amini, Xinyu Yang, Huizi Mao, Daniela Rus, and Song Han. Bevfusion: Multi-task multi-sensor fusion with unified bird's-eye view representation. *arXiv preprint arXiv:2205.13542*, 2022. 2

[16] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2018. 1

[17] 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. *Advances in neural information processing systems*, 32, 2019. 1

[18] Jonah Philion and Sanja Fidler. Lift, splat, shoot: Encoding images from arbitrary camera rigs by implicitly unprojecting to 3d. In *European Conference on Computer Vision*, pages 194–210. Springer, 2020. 1, 2

[19] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015. 1

[20] Hao Shao, Letian Wang, RuoBing Chen, Hongsheng Li, and Yu Liu. Safety-enhanced autonomous driving using interpretable sensor fusion transformer. *arXiv preprint arXiv:2207.14024*, 2022. 2

[21] 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

[22] 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. *Conference on Neural Information Processing Systems (NeurIPS)*, 2022. 1, 2

[23] Yan Yan, Yuxing Mao, and Bo Li. Second: Sparsely embedded convolutional detection. *Sensors*, 18(10):3337, 2018. 1

[24] Zhejun Zhang, Alexander Liniger, Dengxin Dai, Fisher Yu, and Luc Van Gool. End-to-end urban driving by imitating a reinforcement learning coach. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021. 1, 2