

Optimizing Anchor-based Detectors for Autonomous Driving Scenes

Xianzhi Du*
Google

Wei-Chih Hung
Waymo

Tsung-Yi Lin*
Google

Abstract

This paper summarizes model improvements and inference-time optimizations for the popular anchor-based detectors in the scenes of autonomous driving. Based on the high-performing RCNN-RS and RetinaNet-RS detection frameworks designed for common detection scenes, we study a set of framework improvements to adapt the detectors to better detect small objects in crowd scenes. Then, we propose a model scaling strategy by scaling input resolution and model size to achieve a better speed-accuracy trade-off curve. We evaluate our family of models on the real-time 2D detection track of the Waymo Open Dataset (WOD) [24]. Within the 70 ms/frame latency constraint on a V100 GPU, our largest Cascade RCNN-RS model achieves 76.9% AP/L1 and 70.1% AP/L2, attaining the new state-of-the-art on WOD real-time 2D detection. Our fastest RetinaNet-RS model achieves 6.3 ms/frame while maintaining a reasonable detection precision at 50.7% AP/L1 and 42.9% AP/L2.

1. Introduction

Object bounding box detection in autonomous driving scenes is one of the most popular yet challenging tasks in computer vision. Unlike common object detection scenes that cover a wide range of detection scenarios, object scales and object categorizes, autonomous driving scenes typically focus on street driving views where objects of interest are smaller in size and are from less categories. Typical objects of interest for autonomous driving scenes are cars, pedestrians, cyclists, motorists, street signs, etc.

The COCO [17] benchmark has been the de facto benchmark to evaluate performance of object detectors since 2015. As a result, most of the popular object detectors [3–11, 15, 16, 18, 19, 21, 25] are tailored for COCO detection in model design, training recipe, post-processing methods, inference-time optimization, and model scaling strategy. In this work, we aim to optimize common object detectors for autonomous driving. We adopt the RCNN-RS and RetinaNet-RS [7] frameworks as our strong baselines and

*Work done while at Google

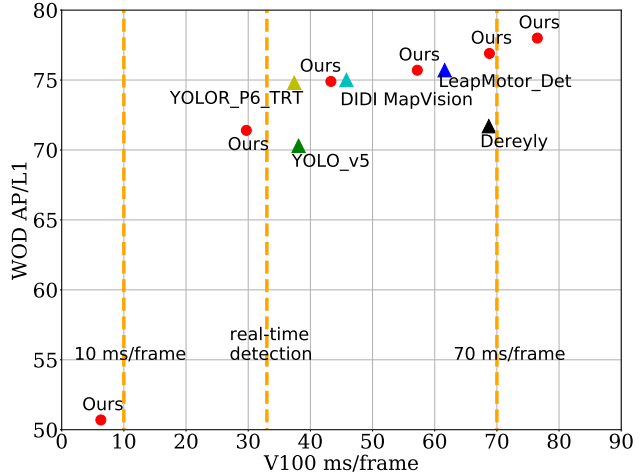


Figure 1. Result comparison on the WOD real-time 2D detection leaderboard. Given the 70 ms/frame latency constraint on a V100 GPU, our best model achieves the new state-of-the-art performance at 76.9% AP/L1. More details can be found in Section 3.

carefully study the effectiveness of tuning common COCO detection settings for autonomous driving scenes in model improvements and inference-time optimizations. Next, we discover a better strategy to scale models in input resolution and backbone scales and propose a family of models that form a better speed-accuracy trade-off curve.

Our family of models are evaluated on the real-time 2D detection track of the Waymo Open Dataset [24] (WOD). We adopt the RCNN-RS models as our main detectors to push for best accuracy and adopt the RetinaNet-RS models to push for fastest speed. Under the 70 frame/ms latency constraint on a V100 GPU, our Cascade RCNN-RS model achieves 76.9% AP/L1 and runs at 68.8 frame/ms, achieving the new state-of-the-art on the real-time 2D detection leaderboard [1]. To further push for highest accuracy and fastest speed, our largest Cascade RCNN-RS model achieves 78.9% AP/L1 and runs at 103.9 ms/frame and our smallest RetinaNet-RS achieves 6.3 ms/frame while maintaining a reasonable AP/L1 at 50.7%, respectively.

Model	Lat (ms/frame)	AP/L1
Faster RCNN-RS	91.8	71.5
+ 3 cascaded heads	122.5	72.8 (+1.3)
+ L2-L6 features	135.1	74.0 (+1.2)
+ Lightweight heads	100.8 (-25%)	73.7
+ 512 proposals	91.7 (-9%)	73.7
+ 0.7 NMS threshold	91.7	74.9 (+1.2)
+ TensorRT&float16	43.3 (-53%)	74.9

Table 1. Ablation studies of the RCNN-RS model improvements and inference-time optimizations on WOD 2D detection. By applying all the changes to the Faster RCNN-RS baseline, the final model achieves +3.4% AP/L1 while being 53% faster.

2. Methodology

2.1. RCNN-RS improvements

2.1.1 Architectural improvements

We adopt the strong RCNN-RS [7] as our main detection framework. RCNN-RS provides improved object detection performance in common detection scenes by adopting modern training techniques and architectural improvements. The modern techniques include scaling jittering augmentation, stochastic depth regularization [13], a longer training schedule and SiLU activation [12, 20]. To further optimize the detection framework for autonomous driving scenes, we make the following changes.

Cascaded heads: The Cascade R-CNN framework [4] shows consistent accuracy improvements over the Faster R-CNN [21] baseline. In this work, we adopt the Cascade RCNN-RS (CRCNN-RS) as our main detection framework. Unlike COCO detection where best accuracy is achieved with two cascaded heads with higher IoUs [7], here we adopt three cascaded detection heads with increasing foreground IoU thresholds {0.5, 0.6, 0.7}.

Lightweight heads: We remove all convolutional layers in the detection heads and the RPN head but only keep the final fully connected layer for bounding box regression and classification. The lightweight head design significantly boosts model speed while achieving similar accuracy as the original head design which consists of 4 convolutional layers and one fully connected layer.

L2-L6 feature pyramid: Detecting objects on a multi-scale feature pyramid is crucial to achieve good performance [15]. A common design choice for COCO detection is to construct a L3-L7 feature pyramid [5, 6, 8, 16, 26]. To better adapt the detector to localize and recognize objects

in smaller scales, we add L2 features to the feature pyramid and remove L7 features.

2.1.2 Inference-time optimizations

Inference-time framework designs: We feed 512 detection proposals instead of 1000 to the second-stage of the CRCNN-RS detector for inference and training. NMS threshold is increased from 0.5 to 0.7 for ROI generation and final detection generation.

Benchmarking improvements: We further adopt the NVIDIA TensorRT optimization with float16 precision to optimize model inference speed.

2.2. RetinaNet-RS improvements

RetinaNet(-RS) [7, 16], a popular anchor-based one-stage object detector, shows competitive detection performance in the low-latency regime for common object detection. In this work, we adopt the RetinaNet-RS framework for our fastest models and apply the changes introduced in Section 2.1 except the ones specific to RCNN-RS. The changes include L2-L6 features, a larger NMS threshold, TensorRT and float16 inference precision.

2.3. Model scaling on WOD

We explore model scaling on the WOD from scaling input resolution and scaling backbone size. A better speed-accuracy trade-off curve is formed by selecting best-performing models within a wide range of computational cost. To scale input resolution, we gradually increase the height of the input image from 384 to 1536 and the width from 640 to 2688. To scale backbone, we adopt architectures at 5 different scales: ResNet-RS-18 \times 0.25¹ [2] (RN18 \times 0.25) that contains 5.3M parameters; SpineNet-49 \times 0.25 (SN49 \times 0.25) that contains 5.6M parameters; SpineNet-49 (SN49) that contains 30.3M parameters; SpineNet-96 (SN96) that contains 37.6M parameters; SpineNet-143 (SN143) that contains 49.7M parameters. Table 2 presents our models of all scales.

3. Experimental Results

3.1. WOD training and eval settings

We conduct experiments on the real-time 2D detection track of the popular Waymo Open Dataset [24]. WOD is a large-scale dataset for autonomous driving that consists of 798 training sequences and 202 validation sequences. Each sequence spans 20 seconds and is densely labeled

¹ \times 0.25 denotes the model’s channel dimension is uniformly scaled to 0.25 of the original size.

Backbone	Input res	Params (M)	FLOPs (B)	Lat (ms/frame)	AP/L1	AP/L2
RN18×0.25	640×1152	5.3	9.3	13.7	59.6	51.3
RN18×0.25	768×1408	5.3	11.4	16.6	62.1	53.9
SN49×0.25	640×1152	5.6	10.4	19.6	63.9	55.5
SN49×0.25	768×1408	5.6	13.0	22.5	66.1	58.1
SN49×0.25	896×1664	5.6	16.0	26.2	67.9	60.0
SN49×0.25	1024×1920	5.6	19.5	30.7	69.2	61.5
SN49	384×640	30.3	-	18.8	62.8	54.0
SN49	512×896	30.3	57	22.0	68.3	59.9
SN49	640×1152	30.3	79	29.7	71.4	63.4
SN49	768×1408	30.3	107	34.9	73.4	65.7
SN49	896×1664	30.3	141	43.3	74.9	67.5
SN49	1024×1920	30.3	180	57.2	75.7	68.6
SN49	1280×2176	30.3	244	68.8	76.9	70.1
SN49	1408×2432	30.3	315	93.8	77.3	70.4
SN49	1536×2688	30.3	372	97.1	77.7	70.9
SN96	1280×2176	37.6	381	76.5	78.0	71.2
SN143	1280×2176	49.7	587	103.9	78.9	72.3

Table 2. **CRCNN-RS model performance on WOD.** We study model scaling on WOD by scaling up input resolution and backbone size. All models adopt the CRCNN-RS framework.

Input res	Lat (ms/frame)	AP/L1	AP/L2
512×896	6.3	50.7	42.9
640×1152	11.6	54.2	46.5
768×1408	13.6	55.5	48.0

Table 3. **RetinaNet-RS performance on WOD.** All models adopt a RN18×0.25 backbone.

at 10 frames per second with camera object detection and tracks.

We train all models on the WOD `train` split with synchronized batch normalization, SGD with a 0.9 momentum rate and a batch size of 256 for 20000 steps on TPUv3 devices [14]. We apply a cosine learning rate schedule with an initial learning rate 0.32. A linear learning rate warm-up is applied for the first 1000 steps. To obtain competitive results, we pretrain our models on the COCO [17] dataset by following the training practices from [7]. Our main results for 2D bounding box detection are reported on the WOD `test` split.

3.2. Improvements from architecture and inference-time optimization

In this section, we show the impact of the RCNN-RS model architectural changes and inference-time model optimizations for autonomous driving scenes. The detailed ablation studies are shown in Table 1. For the architectural

changes, starting from the Faster RCNN-RS model, adding two more cascaded heads increases AP/L1 by **+1.3%**. Introducing L2 features to the multiscale feature pyramid increases AP/L1 by another **+1.2%**. Removing 3×3 convolutional layers in the RPN head and the detection heads speeds up the model by **25%** while achieving similar accuracy. For the inference-time optimizations, by reducing the number of proposals for the second stage from 1000 to 512 and increasing the NMS threshold from 0.5 to 0.7, we further improve AP/L1 by **+1.2%** while being **9%** faster. Finally, optimizing model with TensorRT and changing inference model precision from `float32` to `float16` significantly reduces inference latency by (**53%**).

3.3. Scaling input resolution vs. backbone size

We explore the effectiveness of scaling input resolution vs. scaling backbone size by a grid search over the scales described in Section 2.3 for CRCNN-RS models. The results are presented in Table 2 and Fig. 2. In Fig. 2, we show that within a reasonable input resolution range from height 512 to 1280, scaling input resolution while using a larger backbone is more effective than adopting a smaller backbone with larger input resolutions. To further push for a higher accuracy or a faster speed, scaling backbone size while keeping input resolution becomes a more effective strategy.

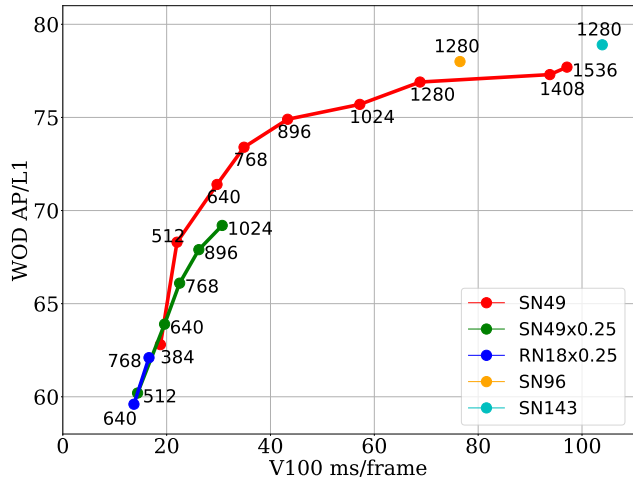


Figure 2. **Model scaling on WOD 2D detection.** We compare CRCNN-RS models adopting SN49, SN96, SN143, SN49 \times 0.25 and RN18 \times 0.25 backbones at various input resolutions. Numbers in this figure represent height of the input image.

3.4. RCNN vs. RetinaNet in the low-latency regime

In this section, we evaluate the performance comparison between RCNN-RS and RetinaNet-RS on WOD. We apply the same training and benchmarking practices. The results are shown in Table 3. We compare the RetinaNet-RS models to the CRCNN-RS models adopting a RN18 \times 0.25 backbone. As shown in Fig. 3, the RetinaNet-RS model underperforms CRCNN-RS by 4.1% AP/L1 while running at a same speed. On the other hand, benefiting from the one-stage framework design, RetinaNet-RS achieves the fastest speed at 6.3 ms/frame.

3.5. WOD Real-time 2D detection results

We present our best performing models and show the performance comparisons to the top-5 entries on the real-time 2D detection leaderboard [1] in Table 4 and Fig. 1. In particular, within the 70 ms/frame constraint, our CRCNN-SN49 model at 1280 \times 2176 input resolution achieves 76.9% AP/L1 and 70.1% AP/L2 and runs at 68.8 ms/frame, outperforming previous best models on the leaderboard. Our CRCNN-SN49 model at 640 \times 1152 input resolution achieves 71.4% AP/L1 and 63.4 AP/L2 and runs at 29.7 ms/frame, achieving real-time object detection while attaining competitive detection precision. In the low-latency regime, our smallest RetinaNet-RS adopting a RN18 \times 0.25 backbone at 512 \times 896 resolution achieves 6.3 ms/frame while maintaining reasonable detection precision at 50.7% AP/L1 and 42.9% AP/L2.

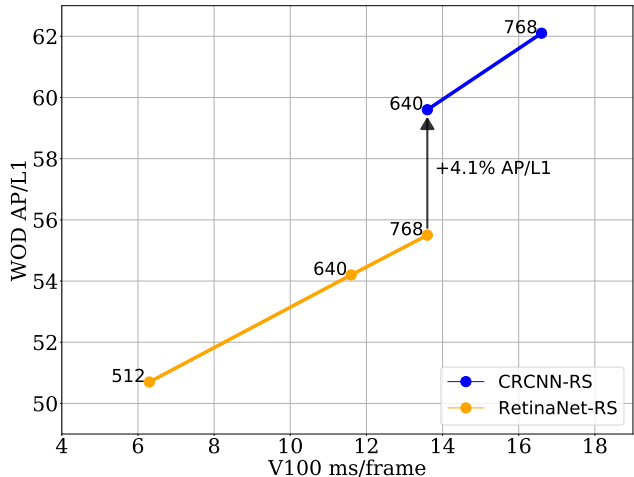


Figure 3. **RetinaNet-RS vs. CRCNN-RS in the low-latency regime.** All models adopt a RN18 \times 0.25 backbone. Numbers in this figure represent height of the input image.

Model	Lat. (ms)	AP/L1	AP/L2
CRCNN-RS-SN49 @640	29.7	71.4	63.4
CRCNN-RS-SN49 @896	43.3	74.9	67.5
CRCNN-RS-SN49 @1024	57.2	75.7	68.6
CRCNN-RS-SN49 @1280	68.8	76.9	70.1
CRCNN-RS-SN96 @1280	76.5	78.0	71.2
LeapMotor_Det [27]	61.6	75.7	70.4
DIDI MapVision [28]	45.8	75.0	69.7
YOLOR_P6_TRT [29]	37.4	74.8	69.6
Deregly_self_ensemble [22]	68.7	71.7	65.7
YOLO_v5 [23]	38.1	70.3	64.1

Table 4. Result comparisons of our models against the top-5 models on the WOD real-time 2D detection leaderboard. We omit results using model ensemble or multiscale test.

4. Conclusion

In this work, we improve the strong two-stage RCNN-RS detector for autonomous driving scenarios from architectural changes and inference-time optimizations. We study the impact of scaling input resolution and model size on the task of WOD real-time 2D detection and propose a family of models for a wide range of latency. We hope this study will help the community to better design detectors for autonomous driving and the optimizations can transfer to more detection frameworks and detection scenarios.

Acknowledgments: We would like to acknowledge Henrik Kretschmar, Drago Anguelov and the Waymo research team for the support. Barret Zoph, Jianwei Xie, Zongwei Zhou, Nimit Nigania for the helpful discussions.

References

- [1] <https://waymo.com/open/challenges/2021/real-time-2d-prediction/>. 1, 4
- [2] Irwan Bello, William Fedus, Xianzhi Du, Ekin D. Cubuk, Aravind Srinivas, Tsung-Yi Lin, Jonathon Shlens, and Barret Zoph. Revisiting resnets: Improved training and scaling strategies, 2021. 2
- [3] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. Yolov4: Optimal speed and accuracy of object detection, 2020. 1
- [4] Zhaowei Cai and Nuno Vasconcelos. Cascade r-cnn: Delving into high quality object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6154–6162, 2018. 1, 2
- [5] Xianzhi Du, Tsung-Yi Lin, Pengchong Jin, Yin Cui, Mingxing Tan, Quoc V. Le, and Xiaodan Song. Efficient scale-permuted backbone with learned resource distribution. In *ECCV*, 2020. 1, 2
- [6] Xianzhi Du, Tsung-Yi Lin, Pengchong Jin, Golnaz Ghiasi, Mingxing Tan, Yin Cui, Quoc V. Le, and Xiaodan Song. Spinenet: Learning scale-permuted backbone for recognition and localization. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11589–11598, 2020. 1, 2
- [7] Xianzhi Du, Barret Zoph, Wei-Chih Hung, and Tsung-Yi Lin. Simple training strategies and model scaling for object detection, 2021. 1, 2, 3
- [8] Golnaz Ghiasi, Tsung-Yi Lin, and Quoc V. Le. Nas-fpn: Learning scalable feature pyramid architecture for object detection. In *CVPR*, 2019. 1, 2
- [9] R. Girshick. Fast r-cnn. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 1440–1448, 2015. 1
- [10] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *2014 IEEE Conference on Computer Vision and Pattern Recognition*, pages 580–587, 2014. 1
- [11] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *ICCV*, 2017. 1
- [12] Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus), 2020. 2
- [13] Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Q Weinberger. Deep networks with stochastic depth. In *ECCV*, 2016. 2
- [14] Norman P. Jouppi, Cliff Young, Nishant Patil, David A. Patterson, Gaurav Agrawal, Raminder Bajwa, Sarah Bates, Suresh Bhatia, Nan Boden, Al Borchers, Rick Boyle, Pierre-luc Cantin, Clifford Chao, Chris Clark, Jeremy Coriell, Mike Daley, Matt Dau, Jeffrey Dean, Ben Gelb, Tara Vazir Ghaemmaghami, Rajendra Gottipati, William Gulland, Robert Hagmann, Richard C. Ho, Doug Hogberg, John Hu, Robert Hundt, Dan Hurt, Julian Ibarz, Aaron Jaffey, Alek Jaworski, Alexander Kaplan, Harshit Khaitan, Andy Koch, Naveen Kumar, Steve Lacy, James Laudon, James Law, Diemthu Le, Chris Leary, Zhuyuan Liu, Kyle Lucke, Alan Lundin, Gordon MacKean, Adriana Maggiore, Maire Mahony, Kieran Miller, Rahul Nagarajan, Ravi Narayanaswami, Ray Ni, Kathy Nix, Thomas Norrie, Mark Omernick, Narayana Penukonda, Andy Phelps, Jonathan Ross, Amir Salek, Emad Samadiani, Chris Severn, Gregory Sizikov, Matthew Snelham, Jed Souter, Dan Steinberg, Andy Swing, Mercedes Tan, Gregory Thorson, Bo Tian, Horia Toma, Erick Tuttle, Vijay Vasudevan, Richard Walter, Walter Wang, Eric Wilcox, and Doe Hyun Yoon. In-datacenter performance analysis of a tensor processing unit. *CoRR*, abs/1704.04760, 2017. 3
- [15] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *CVPR*, 2017. 1, 2
- [16] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *ICCV*, 2017. 1, 2
- [17] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *ECCV*, 2014. 1, 3
- [18] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg. Ssd: Single shot multibox detector. *Lecture Notes in Computer Science*, page 21–37, 2016. 1
- [19] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows, 2021. 1
- [20] Prajit Ramachandran, Barret Zoph, and Quoc V. Le. Searching for activation functions, 2017. 2
- [21] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems*, 2015. 1, 2
- [22] Nikolay Sergievskiy. https://waymo.com/open/challenges/entry/?challenge=DETECTION_2D&emailId=2cd1c01b-4335×tamp=1623340740857905. 4
- [23] Nikolay Sergievskiy. https://waymo.com/open/challenges/entry/?challenge=DETECTION_2D&emailId=2cd1c01b-4335×tamp=1623184567343638. 4
- [24] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, et al. Scalability in perception for autonomous driving: Waymo open dataset. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2446–2454, 2020. 1, 2
- [25] Mingxing Tan, Ruoming Pang, and Quoc V. Le. Efficientdet: Scalable and efficient object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020. 1
- [26] Mingxing Tan, Ruoming Pang, and Quoc V. Le. Efficientdet: Scalable and efficient object detection. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10778–10787, 2020. 2
- [27] Fenfen Wang, Qiankun Xie, Lindong Li, Yaonong Wang, and Hongtao Zhou. https://waymo.com/open/challenges/entry/?challenge=DETECTION_

2D & emailId = 6aee4b36 - 4301 & timestamp = 1623415379657313. 4

- [28] Yueming Zhang, Xiaolin Song, Bing Bai, Tengfei Xing, Chao Liu, Xin Gao, Zihui Wang, Haojin Liao, and Pengfei Xu. [https://waymo.com/open/challenges/entry/?challenge=DETECTION_2D & emailId = 9554504f - c966 & timestamp = 1623399193410618](https://waymo.com/open/challenges/entry/?challenge=DETECTION_2D&emailId=9554504f-c966×tamp=1623399193410618). 4
- [29] Yueming Zhang, Xiaolin Song, Bing Bai, Tengfei Xing, Chao Liu, Xin Gao, Zihui Wang, Haojin Liao, and Pengfei Xu. [https://waymo.com/open/challenges/entry/?challenge=DETECTION_2D & emailId = 9554504f - c966 & timestamp = 1623232203640130](https://waymo.com/open/challenges/entry/?challenge=DETECTION_2D&emailId=9554504f-c966×tamp=1623232203640130). 4