PillarHist: A Quantization-aware Pillar Feature Encoder based on Height-aware Histogram
———— Supplementary Material ————

Sifan Zhou[1†*] Zhihang Yuan[2†] Dawei Yang[2‡] Xing Hu[2] Jian Qian[3] Ziyu Zhao[1]

[1]School of Automation, Southeast University
[2]Houmo AI [3]Fudan University

{sifanjay, hahnyuan, zhaoziyu.950207}@gmail.com,{dawei.yang, xing.hu}@houmo.ai

1. Results on KITTI Test set

Method	3D Detection Easy	Mod.	Hard	mAP	FPS
PointPillars[3]	51.45	41.92	38.89	44.09	63
TANet[6]	53.72	44.34	40.49	46.18	29
PiFeNet [4]	56.39	46.71	42.71	**48.60**	26
PH-PointPillars (Ours)	55.79	45.85	42.15	47.93	**69**

Table 1. Performance of specific PFE modules on the KITTI *test* set in AP_{3D} (R40) for *Pedestrian*.

Comparison with Specific PFE Methods on KITTI test set. Existing methods that focus solely on pillar encoding include TANet [6] and PiFeNet [4], where PiFeNet is tailored specifically for pedestrian detection. To enable a fair comparison, we report the pedestrian category performance on the KITTI test set. We follow the experiment setting of PiFeNet, where the training set is further split into training and validating set with a ratio of 85:15. We select PointPillars as our baseline and only replaced its pillar feature encoding module with our PillarHist to construct **PH-PointPillars** for comparison. As shown in Table 1, our method achieves a 3.84 AP improvement over the baseline (PointPillars) and is 1.75 AP higher than TANet. Compared to PiFeNet, PH-PointPillars achieve comparable performance at 2.7 times the inference speed of PiFeNet, which confirms the effectiveness and efficiency of our PillarHist design. For PiFeNet, although our method's performance was slightly lower (0.73 AP lower), we believe this is because we didn't selecting a better backbone and bounding box regression structure design due to we focuse on pillar feature encoding design. In contrast, the TANet and PiFeNet in Table 1, in addition to their TA module and PAA module design, they also include Coarse-to-Fine Regression (CFR) and Mini-BiFP design for feature encoding and bounding box regression, those extra designs have been shown to be effective than the PointPillar backbone in their paper, respectively.

2. More Ablation Study

Generalization of PillarHist on nuScenes val set. Here, we evaluate our PillarHist's generalization on different pillar-based baselins in nuScenes val set. We selected three representative pillar-based detectors: PointPillars [3], CP-Pillar [12], and PillarNet [9]. In detail, we replace the raw PFE module in each baseline model with our proposed PillarHist, resulting in PH-PointPillars, PH-CP-Pillar, and PH-PillarNet, respectively. As shown in Table 2, the results demonstrate the generalizability of proposed PillarHist, as we observe performance improvements of 1.7 NDS, 1.4 NDS, and 1.6 NDS on the three differnet baselin models, respectively. These results highlight the effectiveness and broad applicability of our PillarHist module in enhancing the representation power of pillar-based 3D object detectors, leading to consistent performance gains across different baseline architectures.

Notably, On the nuScenes dataset, the nuScenes detection score (NDS) is the primary evaluation metric. As shown in Table 2, our proposed method achieves consistent performance gains on the NDS across different pillar-based 3D object detectors. The key contribution that leads to the NDS improvement is the significant reduction in the Mean Average Orientation Error (mAOE) and Mean Average Velocity Error (mAVE), indicating that our method is effective in mitigating the orientation and velocity estimation errors of the detected objects. The improvements can be attributed to the enhanced ability of our PillarHist encoder to capture the spatial and structural information of the point cloud within each pillar. The performance gains on

[*]Work done as an intern at Houmo AI. [†]Equal Contribution. [‡]Corresponding author.

Models	NDS↑	mAP↑	mATE↓	mASE↓	mAOE↓	mAVE↓	mAAE↓
PointPillars [3]	52.1	37.4	0.374	0.266	0.407	0.410	0.202
PH-PointPillars	**53.8** (+1.7)	**38.8** (+1.4)	0.366	0.262	0.387	0.337	0.204
CP-Pillar[12]	56.4	44.9	0.336	0.266	0.532	0.289	0.189
PH-CP-Pillar	**57.8** (+1.4)	**45.8** (+0.9)	0.330	0.264	0.514	0.215	0.188
PillarNet[9]	61.4	52.6	0.296	0.262	0.502	0.250	0.186
PH-PillarNet	**63.0** (+1.6)	**52.7** (+0.1)	0.296	0.256	0.376	0.217	0.182

Table 2. Generalization and quantization performance comparison on nuScene *val* set.

the NDS metric, which reflects the overall detection quality, demonstrate the effectiveness of our proposed PillarHist in improving the robustness and accuracy of pillar-based 3D object detectors on the nuScenes benchmark.

Methods	Car $3D_{R40}$				Ped $3D_{R40}$			
	mAP	Easy	**Mod.**	Hard	mAP	Easy	**Mod.**	Hard
PointPillars [3]	79.98	87.08	77.90	74.97	49.41	54.71	49.01	44.52
PH-PointPillars	81.41	88.80	79.13	76.30	51.07	57.45	50.42	45.36
⇑	**1.43**	**1.72**	**1.23**	**1.33**	**1.66**	**2.74**	**1.41**	**0.84**
CP-Pillar [12]	77.01	84.70	74.46	71.89	42.91	45.74	42.93	39.76
PH-CP-Pillar	77.85	85.62	75.17	72.77	44.93	49.10	44.99	40.72
⇑	**0.84**	**0.92**	**0.71**	**0.88**	**2.02**	**3.36**	**2.06**	**0.96**
PillarNet [9]	80.90	87.21	79.05	76.44	44.51	47.83	44.80	41.01
PH-PillarNet	81.60	88.71	79.05	77.03	46.10	49.75	45.96	42.61
⇑	**0.70**	**1.50**	0.00	**0.59**	**1.62**	**1.92**	**1.16**	**1.60**

Table 3. Comparison with the Pillar-based methods on the KITTI *val* set.

Generalization of PillarHist on KITTI val set. As shown in Table 3, when compared to PointPillars [3], CenterPoint-Pillar [12], and PillarNet [9] methods, our method achieves mAP improvements of 1.43, 0.84, and 0.70 for the vehicle class, respectively. For the pedestrian category, PillarHist achieves even larger performance gains of 1.66, 2.02, and 1.62 points on mAP, respectively. These improvements indicate that our histogram-based point feature encoder can generate more informative features to enhance object detection performance.

bins	Car $3D_{R40}$				Ped $3D_{R40}$			
	mAP	Easy	**Mod.**	Hard	mAP	Easy	**Mod.**	Hard
s=16	80.27	87.57	78.15	75.10	45.79	50.35	45.62	41.41
s=32	80.49	87.98	78.28	75.22	49.79	55.35	49.48	44.56
s=48	81.24	89.31	78.69	75.74	50.08	55.42	49.91	44.93
s=64	**81.41**	88.80	**79.13**	76.30	**51.07**	57.45	**50.42**	45.36
s=80	80.49	87.53	78.44	75.50	49.96	55.82	49.68	44.38

Table 4. Ablation study on the number of bins s on the KITTI *val* set in AP_{3D} (R40).

The Number of Bins Along Height Dimension. Here, we conduct an ablation study on the number of bins s in the height dimension. We take PointPillars as our baseline. Notably, PointPillars limits the maximum number of points as 32 in each pillar. We vary the number of bins from 16 to 80, and the results are presented in Table 4. As the number of bins increased, the performance of the model improved steadily. However, the performance does not exhibit a monotonic improvement trend and exhibit a degradation when the number of bins was increased to 80. Although larger number of bins can better encode discriminative features in the height dimension, the performance saturates when the number of bins becomes excessively large. Based on these experimental results, we selected a value of 64 for the number of bins.

3. Quantization Preliminaries

Quantization for tensor. The quantization operation is defined as the mapping of a floating-point (FP) value x (weights or activations) to an integer value x_{int} according to the following equation:

$$x_{int} = clamp(\lfloor \frac{x}{s} \rceil + z, q_{min}, q_{max}) \tag{1}$$

where $\lfloor \cdot \rceil$ is the rounding-to-nearest operator, which results in the rounding error Δr. The function $clamp(\cdot)$ clips the values that lie outside of the integer range $[q_{min}, q_{max}]$, incurring a clipping error Δc. x_{int} represents the quantized integer value. z is zero-point. s denotes the quantization scale factor, which reflects the proportional relationship between FP values and integers. $[q_{min}, q_{max}]$ is the quantization range determined by the bit-width b. Here, we adopt uniform signed symmetric quantization, as it is the most widely used in TensorRT [7] and brings significant acceleration effect. Therefore, $q_{min} = -2^{b-1}$ and $q_{max} = 2^{b-1} - 1$. Nonuniform quantization [2] is challenging to deploy on hardware, so we disregard it in this work. Generally, weights can be quantized without any need for calibration data. Therefore, the quantization of weights is commonly solved using grid search or analytical approximations with closed-form solution [8] to minimize the mean squared error (MSE) in PTQ. However, activation quantization is input-dependent, so often requires a few batches of calibration data for the estimation of the dynamic ranges to converge. To approximate the real-valued input x, we perform the de-quantization step:

$$\hat{x} = (x_{int} - z) \cdot s \tag{2}$$

where $\hat{x}$ is the de-quantized FP value with an error that is introduced during the quantization process.

Quantization range. If we want to reduce clipping error Δc, we can increase the quantization scale factor s to expand the quantization range. However, increasing s leads to increased rounding error Δr because Δr lies in the range $\left[-\frac{s}{2}, \frac{s}{2}\right]$. Therefore, the key problem is how to choose the quantization range (x_{min}, x_{max}) to achieve the right trade-off between clipping and rounding error. Specifically, when we set fixed bit-width b, the quantization scale factor s is determined by the quantization range:

$$s = (x_{max} - x_{min}) \,/\, \left(2^b - 1\right) \tag{3}$$

Max-min calibration. We can define the quantization range as:

$$x_{max} = max(|x|), x_{min} = -x_{max} \tag{4}$$

to cover the whole dynamic range of the floating-point value x. This leads to no clipping error. However, this approach is sensitive to outliers as strong outliers may cause excessive rounding errors.

Quantization for network. Following previous methods [5, 14], for a float model with N layer, we primarily focus on the quantization of convolutional layers or linear layers, which mainly involves the handling of weights and activations. For a given layer L_i, we initially execute quantization operations on its weight and input tensor, as illustrated in Eq 7 and 2, yielding $\hat{W}_i$ and $\hat{I}_i$. Consequently, the quantized output of this layer can be expressed as follows.

$$\hat{A}_i = f(BN(\hat{I}_i \circledast \hat{W}_i) \tag{5}$$

where $\circledast$ denotes the convolution operator, $BN(\cdot)$ is the Batch-Normalization procedure, and $f(\cdot)$ is the activation function. Quantization works generally take into account the convolution, Batch Normalization (BN), and activation layers.

Weight/Activations Quantization. Specifically, for a weight or activation tensor X, firstly obtain the x_{max} and x_{min} according to Eq.4, and calculate the initial quantization parameter s_0 following Eq. 8. Then linearly divide the interval $[\alpha s_0, \beta s_0]$ into T candidate bins, denoted as $\{s_t\}_{t=1}^{T}$. α, β and T are designed to control the search range and granularity. Finally, search $\{s_t\}_{t=1}^{T}$ to find the optimal s_{opt} that minimizes the quantization error.

$$\text{argmin}_{s_t} \|(X - \hat{X}(s_l))\|_F^2 \tag{6}$$

$\|\cdot\|_F^2$ is the Frobenius norm (MSE Loss).

Gird Search for Weight and Activation Quantization Scale. For a weight or activation tensor X, we can get their initial quantization scale factor using the following equation:

$$\hat{x} = (clamp(\lfloor \frac{x}{s} \rceil + z, q_{min}, q_{max}) - z) \cdot s \tag{7}$$

Algorithm 1: Grid search

Input: the input of full precision tensor X, bit-width b and T bins.
Output: scale factor s_{opt} with $min(\|(X - \hat{X}(s_l))\|_F^2)$.

using $x_{max} = max(|x|)$ get max value of tensor X
set $range = x_{max}$, $c_{best} = 100$
set $v_{min} = x_{min}$ and $v_{max} = x_{max}$
for *i in range(1, T)* **do**
  threshold = $range/T/i$
  x_{min} = $-threshold$, x_{max} = $threshold$
  get scale s_t with x_{min} and x_{max} using Eq 8
  input the quantized value $\hat{x}$ and FP value x using Eq 7 to get score c
  update v_{min} and v_{max} when $c < c_{best}$ and update c_{best} = c
get v_{min} and v_{max} with the minimal score c
get final scale s_{opt} with v_{min} and v_{max} using Eq 8 scale s_{opt}

$$s = (x_{max} - x_{min}) \,/\, \left(2^b - 1\right) \tag{8}$$

$$\text{argmin}_{s_t} \|(X - \hat{X}(s_l))\|_F^2 \tag{9}$$

$\|\cdot\|_F^2$ is the Frobenius norm (MSE Loss). Refer to appendix for more details about grid search. Then linearly divide the interval $[\alpha s_0, \beta s_0]$ into T candidate bins, denoted as $\{s_t\}_{t=1}^{T}$. α, β and T are designed to control the search range and granularity. Finally, search $\{s_t\}_{t=1}^{T}$ to find the optimal s_{opt} that minimizes the quantization error, the grid search method in Algorithm 2.

Naive Post-Training Quantization Algorithm. The naive post-training quantization algorithm are as follows, the grid search method in Algorithm 2.

4. Dataset and Implementation Details

KITTI Dataset. The KITTI dataset uses a LiDAR with 64 channels. The KITTI has 7481 training samples and 7518 test samples. The training samples are split into a $train$ set with 3,717 samples and a val set with 3,769 samples following the common setting [3, 11]. The detected boxes are classified into three subsets: "Easy", "Moderate" and "Hard" based on the levels of difficulty. We report Mean Average Precision (mAP).

nuScens Dataset. The nuScenes [1] dataset contains 700 training scenes, 150 val scenes and 150 test scenes. Each frame is generated approximately 30K points by a 32 channels LiDAR sampled with 20Hz. The training, validation and testing set have 28K, 6K and 6K annotated keyframes, respectively, It contains 10 categories in total. Its main

Algorithm 2: PillarHist Naive Post-Training Quantization

Input: Pretrained FP model with N layers; Calibration dataset D^c.

Output: quantization parameters of both activation and weight in network, i.e., weight scale s_w, weight zero-point z_w, activation scale s_a and activation zero-point z_a.

Get only weight quantization parameters s_w and z_w to minimize Eq 9 in every layer using the grid search algorithm; **for** $L_n = \{L_i | i = 1, 2, ...N\}$ **do**

  Optimize only activation quantization parameters s_a and z_a to minimize Eq 9 in layer L_i using the grid search algorithm;

  Quantize layer L_i with the quantization parameters s_w, z_w, s_a, and z_a;

ranking metric is the nuScenes detection score (NDS). The Mean Average Precision (mAP) was also used based on distances between the centers of the predictions and ground-truths on the bird-eye view at thresholds 0.5, 1, 2, and 4m.

Waymo Open Dataset. The Waymo Open Dataset[10] contains 1150 sequences in total, 798 for training, 202 for validation and 150 for test. Each sequence is sampled at 10Hz with a 64 channels LiDAR containing 6.1M vehicle, 2.8M pedestrian, and 67k cyclist boxes. Each frame covers a scene with a size of 150m×150m. The official evaluation tools evaluated the methods in two difficulty levels: LEVEL1 for boxes with more than five LiDAR points, and LEVEL2 for boxes with at least one LiDAR point.

Training Details Following previous methods [12, 13], we utilize the Adam optimizer with one-cycle learning rate policy. The initial learning rate is set to 10e-4 during training. The learning rate gradually increases to 0.001 in the first 50% epochs and then gradually decreases to 10e-5 for the remaining training process. We set the weight decay to 0.01 and the momentum to a range of 0.85 to 0.95, For data augmentation, the whole point cloud is flipped randomly along the X axis, randomly is rotated along the Z axis in the range $[-\pi/4, \pi/4]$, and globally is scaled by a random factor sampled from $[0.95, 1.05]$. Regarding the pillar size, network architecture and loss function. Our experiments are executed on 4 Nvidia Tesla V100 GPUs.

Acknowledgments

This work was supported by National Natural Science Foundation of China (No.62271143), Frontier Technologies RD Program of Jiangsu (No. BF2024060) and the Big Data Computing Center of Southeast University.

References

[1] Holger Caesar, Varun Bankiti, Alex Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. Nuscenes: A multimodal dataset for autonomous driving. In *CVPR*, pages 11621–11631, 2020. 3

[2] Yongkweon Jeon, Chungman Lee, Eulrang Cho, and Yeonju Ro. Mr. biq: Post-training non-uniform quantization based on minimizing the reconstruction error. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12329–12338, 2022. 2

[3] Alex H Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. Pointpillars: Fast encoders for object detection from point clouds. In *CVPR*, pages 12697–12705, 2019. 1, 2, 3

[4] Duy Tho Le, Hengcan Shi, Hamid Rezatofighi, and Jianfei Cai. Accurate and real-time 3d pedestrian detection using an efficient attentive pillar network. *IEEE Robotics and Automation Letters*, 8(2):1159–1166, 2022. 1

[5] Jiawei Liu, Lin Niu, Zhihang Yuan, Dawei Yang, Xinggang Wang, and Wenyu Liu. Pd-quant: Post-training quantization based on prediction difference metric. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 24427–24437, 2023. 3

[6] Zhe Liu, Xin Zhao, Tengteng Huang, Ruolan Hu, Yu Zhou, and Xiang Bai. Tanet: Robust 3d object detection from point clouds with triple attention. In *AAAI*, pages 11677–11684, 2020. 1

[7] Szymon Migacz. 8-bit inference with tensorrt. In *GPU technology conference*, page 5, 2017. 2

[8] Markus Nagel, Marios Fournarakis, Rana Ali Amjad, Yelysei Bondarenko, Mart Van Baalen, and Tijmen Blankevoort. A white paper on neural network quantization. *arXiv preprint arXiv:2106.08295*, 2021. 2

[9] Guangsheng Shi, Ruifeng Li, and Chao Ma. Pillarnet: Real-time and high-performance pillar-based 3d object detection. In *ECCV*, 2022. 1, 2

[10] 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 *CVPR*, pages 2446–2454, 2020. 4

[11] Yan Yan, Yuxing Mao, and Bo Li. Second: Sparsely embedded convolutional detection. *Sensors*, 18(10):3337, 2018. 3

[12] Tianwei Yin, Xingyi Zhou, and Philipp Krahenbuhl. Center-based 3d object detection and tracking. In *CVPR*, pages 11784–11793, 2021. 1, 2, 4

[13] Sifan Zhou, Zhi Tian, Xiangxiang Chu, Xinyu Zhang, Bo Zhang, Xiaobo Lu, Chengjian Feng, Zequn Jie, Patrick Yin Chiang, and Lin Ma. Fastpillars: A deployment-friendly pillar-based 3d detector. *arXiv preprint arXiv:2302.02367*, 2023. 4

[14] Sifan Zhou, Liang Li, Xinyu Zhang, Bo Zhang, Shipeng Bai, Miao Sun, Ziyu Zhao, Xiaobo Lu, and Xiangxiang Chu. Lidar-ptq: Post-training quantization for point cloud 3d object detection. *International Conference on Learning Representations (ICLR)*, 2024. 3