Towards Efficient 3D Object Detection with Knowledge Distillation

Jihan Yang[1] **Shaoshuai Shi**[2] **Runyu Ding**[1] **Zhe Wang**[3] **Xiaojuan Qi**[1]
[1]The University of Hong Kong [2]Max Planck Institute for Informatics [3]SenseTime Research
{jhyang, ryding, xjqi}@eee.hku.hk, {shaoshuaics, wzlewis16}@gmail.com

Abstract

Despite substantial progress in 3D object detection, advanced 3D detectors often suffer from heavy computation overheads. To this end, we explore the potential of knowledge distillation (KD) for developing efficient 3D object detectors, focusing on popular pillar- and voxel-based detectors. In the absence of well-developed teacher-student pairs, we first study how to obtain student models with good trade offs between accuracy and efficiency from the perspectives of model compression and input resolution reduction. Then, we build a benchmark to assess existing KD methods developed in the 2D domain for 3D object detection upon six well-constructed teacher-student pairs. Further, we propose an improved KD pipeline incorporating an enhanced logit KD method that performs KD on only a few pivotal positions determined by teacher classification response, and a teacher-guided student model initialization to facilitate transferring teacher model's feature extraction ability to students through weight inheritance. Finally, we conduct extensive experiments on the Waymo and KITTI dataset. Our best performing model achieves 65.75% LEVEL 2 mAPH, surpassing its teacher model and requiring only 44% of teacher flops on Waymo. Our most efficient model runs 51 FPS on an NVIDIA A100, which is $2.2\times$ faster than PointPillar with even higher accuracy on Waymo. Code is available at https://github.com/CVMI-Lab/SparseKD.

1 Introduction

3D object detection from point clouds is a fundamental perception task with broad applications on autonomous driving, robotics and smart city, etc. Recently, benefited from large-scale 3D perception datasets [12, 2, 45] and advanced point- [36], pillar- [24, 51] and voxel-based [13, 60] representations of sparse and irregular LiDAR point cloud scenes, 3D detection has achieved remarkable progress [55, 43, 41, 1, 58]. However, stronger performance is often accompanied with heavier computation burden (see Figure 1), rendering their adoption in real-world applications still a challenging problem.

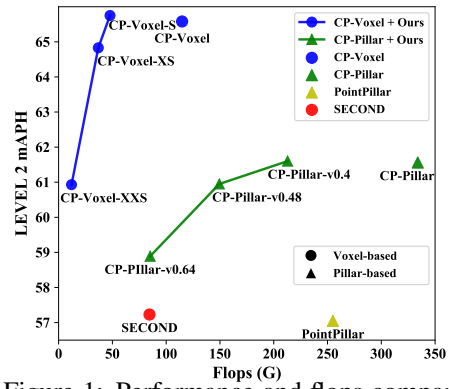


Figure 1: Performance and flops comparison of single-stage detectors on Waymo: CP-Pillar [58], CP-Voxel [58], SECOND [55], PointPillar [24] and Ours.

Recent attempts to improve efficiency focus on developing specified architectures for point-based 3D object detectors [6, 59], not generalizable to a wide spectrum of pillar/voxel-based methods [60, 24, 55, 41, 58, 10]. Here, we aim at a model-agnostic framework for obtaining efficient and accurate 3D object detectors with knowledge distillation (KD). Due to its effectiveness, generality and simplicity, KD has become a de facto strategy to develop efficient models in a variety of 2D tasks [18, 30, 9, 19, 54]. It facilitates

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

improving the performance of a lightweight and efficient student model by harvesting knowledge learned by an accurate yet computationally heavy teacher model. Despite of numerous studies in 2D tasks, the investigation of KD for efficient 3D object detection has largely escaped research attention with unresolved research challenges. In this paper, we conduct the first systematic study on knowledge distillation for developing high-performance and efficient 3D LiDAR-based detectors.

First, we study *how to obtain lightweight student detectors with satisfactory efficiency and accuracy trade offs given a pre-trained teacher 3D object detector*. Different from the 2D domain where well-developed backbone architectures with different model efficiencies are readily available (e.g., ResNet 18 *vs.* ResNet 50) [44, 47, 16], such scalable backbones in 3D scenes are still under-explored. This makes the design of suitable student models a non-trivial problem. Intuitively, a good student model should achieve a good compromise between accuracy and efficiency as a poor student model may have inferior capabilities or architectural level shortcomings, causing difficulties for further knowledge distillation. In this perspective, we first propose the Cost Performance Ratio (CPR) to fairly evaluate a model in terms of both efficiency and capability. Then, we study different factors including model width (*i.e.* number of channels), depth (*i.e.* number of layers) and input resolution on both pillar-based and voxel-based architectures. Specifically, we find that pillar-based architecture favors input-level compression (*i.e.* reduce input resolution) while voxel-based detector prefers width-level compression, due to less spatial redundancy of voxel-based detectors.

Second, we empirically investigate *the effectiveness of existing knowledge distillation methods on this new setting* upon accurate teacher models and efficient student models. As there is no prior research for this problem, we benchmark seven existing knowledge distillation methods on top of six teacher-student pairs covering voxel- and pillar-based architectures. Specifically, we evaluate the following major streams: **logit KD** distilling on model outputs (KD [18] and GID-L [9]), **feature KD** mimicking intermediate features (FitNet [40], Mimic [26], FG [50] and GID-F [9]) and **label KD** leveraging teacher's predictions for label assignment [35]. Further, we also study their synergy effects and empirically find that feature KD outperforms others individually, but fails to cooperate with other KD manners to further boost the performance in 3D detection. This is potentially caused by the fact that logit and label KD can add an implicit regularization to the intermediate features. The best strategy that we obtain through empirical studies is to use FG [50] or Mimic [26] individually.

Third, we propose *simple, general, and effective strategies to improve knowledge distillation on 3D object detection* upon the strong KD baseline derived as above. Motivated by the extreme imbalance between small informative areas containing 3D objects and large redundant background areas in 3D scenes, we design a modified logit KD method, namely pivotal position logit KD, enforcing imitation on only locations with highly confident or top-ranked teacher predictions. These areas are shown to be near instance centers or error-prone positions. Besides, to facilitate effective knowledge transfer from teacher to student, we develop Teacher Guided Initialization (TGI) which remaps pre-trained teacher parameters to initialize a student model. This is shown to be effective in inheriting teacher models' feature extraction abilities while collaborating well with logit and label KD techniques.

Finally, our empirical studies on efficient model design and knowledge distillation methods yield superior performance in delivering efficient and effective pillar- and voxel-based 3D detectors. This is extensively verified on the largest annotated 3D object detection dataset – Waymo [45]. As shown in Figure 1, our best performing model, *i.e.* CP-Voxel-S, even outperforms its teacher model (*i.e.* CP-Voxel) while has $2.4\times$ fewer flops. Moreover, our most efficient pillar-based model (*i.e.* CP-Pillar-v0.64) can run 51 FPS with 58.89% mAPH while previous fastest voxel/pillar-based detector – PointPillar runs 23 FPS with 57.03% mAPH on an NVIDIA A100 GPU (see Sec. D). Our method is also shown to be effective in knowledge transfer from heavy two-stage object detectors to lightweight single-stage detectors. In addition, our method can generalize well to other settings such as KITTI dataset with SECOND as well as advance compression methods in Sec. 5.5, other detectors in Sec. F, and even 3D semantic segmentation in Sec. E.

2 Related Work

3D LiDAR-based Object Detection targets to localize and classify 3D objects from point clouds. Point-based methods [43, 6, 56, 59] took raw point clouds and leveraged PointNet++ [36] to extract sparse point features and generate point-wise 3D proposals. Pillar-based works [24, 51] finished voxelization in bird eye's view and extracted pillar-wise features with PointNet++. Voxel-based methods [60, 55, 41] voxelized point clouds and obtained voxel-wise features with 3D sparse convolutional networks, which is the most popular data treatment. Besides, range-based works [1, 46]

were proposed for long-range and fast detection. Recently, designing efficient 3D detectors has drawn some attentions [6, 59] with raw point data treatment. In this work, we focus on exploring model-agnostic knowledge distillation methods to boost the efficiency of 3D detectors.

Knowledge Distillation aims to transfer knowledge from a large teacher model to a lightweight student network, which is a thriving area in efficient deep learning. Hinton *et al.* proposed the seminal concept of knowledge distillation (KD) [18], which distilled knowledge between teacher and student on the output level (*i.e.* prediction logits). Another line of research proposed to help student's optimization with hints stored in informative intermediate features from teacher [40, 20, 23, 17, 22, 5]. In addition, some works attempted distillation techniques in 2D object detection [26, 50, 9, 37, 57, 35] by emphasizing instance-wise distillation and feature knowledge. Mimic [26], FG [50] and GID [9] sampled local region features with box proposals or custom indicators for foreground-aware feature imitation. Label KD [35] utilized teacher's information for label assignment of student. Recently, knowledge distillation has also been leveraged to transfer knowledge in multi-modality setup [14, 31] or multi-frame to single-frame setup [52] in 3D detection area. However, to the best of our knowledge, we are the first to explore knowledge distillation in the most popular setup: single-frame 3D LiDAR-based object detection. In this work, we propose an enhanced 3D detection KD pipeline with our designed efficient 3D detectors on the popular voxel/pillar data representation. Our lightweight detectors CP-Voxel-S and CP-Pillar-v0.4 slightly outperform their state-of-the-art teacher detectors separately while requiring much less computation overhead.

3 Designing Efficient Student Networks

As there are no readily available lightweight backbone architectures for constructing student networks, we carry out an empirical study on how to obtain an efficient model with satisfactory efficiency and accuracy trade offs to facilitate further knowledge distillation. In this section, we will first describe our experimental setups and model evaluation metrics. Then, we study different strategies to obtain an efficient model and conduct an in-depth analysis on how to achieve good trade offs for pillar- and voxel-based architectures.

3.1 Basic Setups and Evaluation Metrics

Basic setups. For detector architectures, we focus on two variants of the state-of-the-art model CenterPoint [58]: CenterPoint-Pillar (CP-Pillar) and CenterPoint-Voxel (CP-Voxel), covering the most popular pillar- and voxel-based 3D detectors [55, 60, 24, 51, 41, 42, 10]. For dataset, we perform all experiments on the largest annotated 3D LiDAR perception dataset Waymo Open Dataset (WOD) [45] with 20% training samples for fast verification. For model training, we follow the training scheme of popular 3D detection codebase OpenPCDet [49] to ensure fair comparisons and standardization. Note that there is no any knowledge distillation method engaged in this stage.

Evaluation metrics. Following [38, 27], we employ number of parameters, flops, activations, latency (*i.e.* test time) and peak GPU training memory (batch size 1) as quantitative indicators to evaluate model efficiency from parameter, computation and memory throughout aspects. Note that activations, flops and latency are averaged over 99 frames with a GTX-1060 GPU. We present latency more for reference since it largely depends on hardware devices as well as operation-level optimizations (see Sec. D). We use LEVEL 2 mAPH as the performance evaluation metric following WOD [45].

Since our major target here is to design student networks with favorable trade offs between performance and efficiency, we propose a quantitative indicator, namely Cost Performance Ratio (CPR), to directly measure a model in this respective. To construct CPR, we use activations (acts) as the metric to evaluate efficiency since it strongly correlates with the runtime on hardware accelerators such as GPUs as shown in [38]. Our CPR finally combines the activation ratio and mAPH as follows:

$$\text{CPR} = 0.5 \times (1 - \frac{\text{acts}_s}{\text{acts}_t}) + 0.5 \times (\frac{\text{mAPH}_s}{\text{mAPH}_t})^3, \tag{1}$$

where the subscripts s and t represent student and teacher model respectively. CPR is normalized to $[0, 1]$ by weighting the relative activation decrease and performance drop ratio of a student network compared to the teacher. Notice that the third power is used for the performance degradation term to penalize acceleration methods that result in drastic performance degradation more severely. We argue that it is necessary to ensure a relative good performance of efficient detectors as models with poor accuracy may suffer from architectural level problems which can cause difficulties in developing knowledge distillation techniques and prohibiting obtaining accurate and efficient detectors.

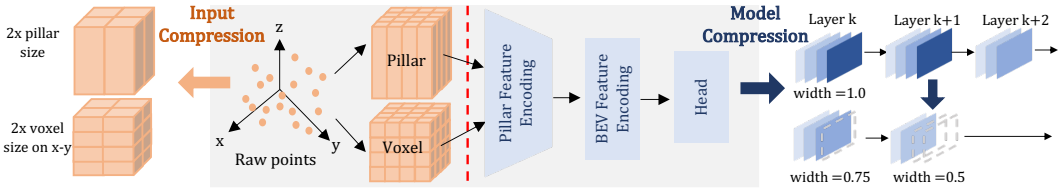


Figure 2: Architecture of detector and illustration of model (right) and input (left) compression.

3.2 Acceleration Strategy

Here, to obtain efficient models, we investigate model and input resolution compression techniques on pillar- and voxel-based 3D detectors as shown in Figure 2 and study their impacts on model accuracy.

Model Compression. Here, we compress a given teacher backbone by trimming it along depth (*i.e.* number of layers) or width (number of channels) as shown in the right of Figure 2. To relieve the burden of exhaustive layer-wise studies, we leverage the functional similarity among different layers to group them into three major modules: Pillar Feature Encoding (PFE) module, Bird eye's view Feature Encoding (BFE) module and detection head (see Figure 2: right), and conduct analysis on module-level. Specifically, PFE corresponds to network components before projecting features to bird eye's view (BEV) grid, including sparse 3D convolutional backbone [13] to extract voxel-wise features in voxel-based architectures or PointNet++ [36] to encode pillar-wise features from points in pillar-based detectors. After PFE, the features are aggregated to pillar features and mapped to 2D BEV grid. Then, the BFE consisting of 2D convolutional layers is adopted to extract final detection feature map on the BEV grid. Finally, the detection head takes outputs from BFE to produce final prediction results. For width pruning, we consider the width of teacher detectors as 1.0, and slim each module of the student by decreasing its number of channels with a given width. Depth trimming also follows this paradigm but still maintains the minimal structure of each module (*e.g.* downsample and upsample layers) to ensure basic detection ability. Model trimming results are shown in Table 1.

Table 1: Model compression results. Teacher models are marked in gray. See text for details.

Detector		Architecture: Width PFE	BFE	Head	Depth PFE	BFE	Efficiency: Params (M)	Flops (G)	Acts (M)	Latency (ms)	Mem. (G)	LEVEL 2 mAPH	CPR
CP-Pillar		1.00	1.00	1.00	1.00	1.00	5.2	333.9	303.0	157.9	5.2	59.09	-
	(a)	1.00	0.50	0.50	1.00	1.00	1.3	87.6	161.8	78.5	3.2	54.50	0.63
	(b)	0.50	0.50	0.50	1.00	1.00	1.3	85.0	152.7	74.0	2.9	52.33	0.60
	(c)	1.00	1.00	1.00	1.00	0.50	2.2	258.5	234.1	118.5	4.3	55.24	0.52
	(d)	1.00	1.00	1.00	1.00	0.33	1.4	234.6	210.0	107.8	4.0	47.97	0.42
CP-Voxel		1.00	1.00	1.00	1.00	1.00	7.8	114.7	101.9	125.7	2.8	64.29	-
	(a)	1.00	0.50	0.50	1.00	1.00	4.0	47.8	65.7	98.0	2.1	62.23	0.63
	(b)	0.75	0.50	0.50	1.00	1.00	2.8	36.9	58.4	88.2	1.9	61.16	0.64
	(c)	0.50	0.50	0.50	1.00	1.00	1.9	28.8	51.2	75.1	1.7	59.47	0.64
	(d)	0.50	0.25	0.25	1.00	1.00	1.0	12.0	33.1	70.4	1.3	56.26	0.67
	(e)	0.25	0.25	0.25	1.00	1.00	0.5	7.3	25.8	66.0	1.1	49.84	0.61
	(f)	1.00	1.00	1.00	0.50	0.50	3.0	63.9	65.2	73.0	1.9	60.95	0.61
	(g)	1.00	1.00	1.00	0.33	0.33	1.8	47.9	52.2	59.0	1.6	55.78	0.57

Input Compression. Besides model complexities, input resolution also has impacts on model efficiency [48, 37]. For instance, by halving the input resolution, the computation overhead for 2D convolution layers in the BFE module and detection head will be reduced to $\frac{1}{4}$ (see Figure 2: left). Besides, there are large background areas in the sparse and large-scale 3D scenes, which naturally has redundancies and offers the possibility of processing the data on a coarser resolution for input compression. Specifically, input compression is realized by increasing the voxel/pillar size on the x-y plane when constructing voxel/pillar. As shown in Table 2, we gradually increase students' voxel size with 25% of teachers' voxel size, and record their efficiency and accuracy metrics.

3.3 Conclusion and Analysis

By analyzing the experimental results shown in Table 1 and Table 2, we draw the following conclusions on efficient model design for pillar- and voxel-based detectors.

Width *vs.* depth compression: width-level pruning is preferred. As shown in Table 1, trimming networks on width generally achieves higher CPR than on depth for both CP-Pillar and CP-Voxel. For instance, with stronger performance, CP-Voxel (d) needs $1.5\times$ fewer acts and $4\times$ fewer flops compared to CP-Voxel (g). As backbones of 3D detectors are much shallower than their 2D counterparts

Table 2: Input compression results. Teacher models are marked in gray. See text for details.

Architecture		Efficiency					LEVEL 2 mAPH	CPR
Detector	Voxel Size (m)	Params (M)	Flops (G)	Acts (M)	Latency (ms)	Mem. (G)		
CP-Pillar	0.32	5.2	333.9	303.0	157.9	5.2	59.09	-
	0.40	5.2	212.9	197.7	103.4	3.8	57.55	0.64
	0.48	5.2	149.4	142.3	81.9	3.0	56.27	0.70
	0.56	5.2	109.9	109.0	66.3	2.6	54.45	0.71
	0.64	5.2	85.1	88.0	54.5	2.1	52.81	0.71
CP-Voxel	0.100	7.8	114.8	101.9	125.7	2.8	64.29	-
	0.125	7.8	77.5	70.1	99.9	2.2	61.55	0.59
	0.150	7.8	53.9	50.0	84.3	1.8	58.14	0.62
	0.175	7.8	44.3	41.2	74.1	1.5	55.99	0.63
	0.200	7.8	32.9	31.4	67.5	1.3	52.80	0.62

(*e.g.* only 19 convolution layers in CP-Pillar and 35 convolution layers in CP-Voxel), this renders depth compression more challenging and less scalable than width compression in 3D detection.

Module-wise pruning selection: PFE module has the least redundancy to be reduced. Since PFE, BFE and detection head perform different functions, they might have their own redundancies in 3D detection. Comparing CP-Voxel (c), (d) and (e) as well as CP-Pillar (a) and (b) in Table 1, we find that less pruning on PFE achieves significantly higher CPR, demonstrating that network parameters in the PFE module is crucial for high performance and it is necessary to maintain network complexities in PFE modules.

Favorable compression strategies for different detection architectures. Comparing results of different compression strategies in Table 1 and 2, we find that pillar-based architecture (*i.e.* CP-Pillar) is more suitable for input compression while voxel-based architecture (*i.e.* CP-Voxel) prefers width-level compression. This difference mainly lies in different spatial redundancy of BEV features on these two detectors. Since the PFE module of voxel-based detector downsamples input resolution to $1/8$, the $0.1m$ input voxel size will be magnified to $0.8m$ on BEV features. Hence, less spatial redundancy could be further compressed for CP-Voxel. However, CP-Pillar often keeps the same resolution between input and BEV features on Waymo [49, 58], and thus has larger resolution redundancy on BEV grid, which facilitates designing student network with coarser input resolution.

Summarized student networks. Driven by the above analysis, we finally derive the following compressed student models with good trade offs between efficiency and performance. For CP-Pillar, as it is friendly to input compression, we adopt the input-compressed models with voxel size 0.40, 0.48 and 0.64 in Table 2, named **CP-Pillar-v0.4**, **CP-Pillar-v0.48** and **CP-Pillar-v0.64**, respectively. As for CP-Voxel, we choose its width compressed models (a) (b) and (d) in Table 1, named **CP-Voxel-S**, **CP-Voxel-XS** and **CP-Voxel-XXS**, respectively.

4 Benchmark Knowledge Distillation for 3D Object Detection

With student networks derived in Section 3, we are now ready to conduct our empirical study on knowledge distillation for 3D object detection. Here, we benchmark seven popular 2D KD methods including logit KD (*i.e.* KD [18] and GID-L [9]), feature KD (*i.e.* FitNet [40], Mimic [26], FG [50] and GID-F [9]) and label KD [35] on six teacher-student pairs with comprehensive analysis. Notice that GID and label KD are state-of-the-art 2D KD detection methods, and GID is divided into GID-L and GID-F to investigate logit KD and feature KD separately. Implementation details and the value of hyper-parameters are described in the Sec. B.

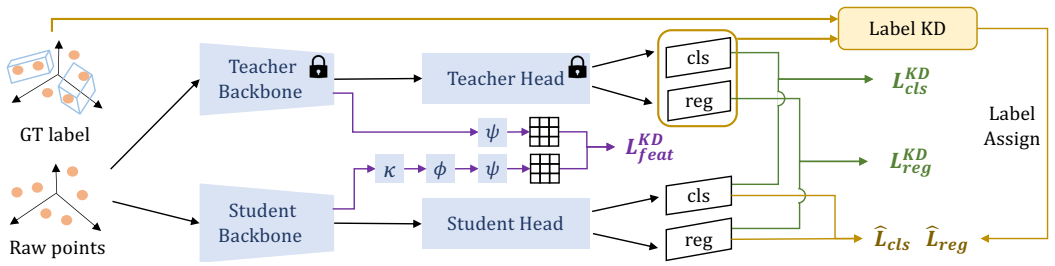


Figure 3: Overall KD paradigm for 3D detection. Teacher weights are frozen during the whole distillation procedure. Logit, feature and label KD are colored by green, purple and yellow, respectively.

4.1 Paradigm

We first evaluate existing 2D distillation methods for 3D detection. As shown in Figure 3, our overall KD paradigm for 3D detection contains three parts: logit, feature and label KD to leverage teacher guidance in response, intermediate feature and label assignment levels, respectively.

Logit KD is the most classical distillation approach introduced by [18]. It takes teacher model's final response as guidance for training a student network and is closely related to the specific task. In 3D object detection, we calculate the logit KD loss between teacher and student outputs as follows:

$$\mathcal{L}_{\text{cls}}^{\text{KD}} = \mathbb{E}[m_{\text{cls}} \| \kappa(p_{\text{cls}}^{s}) - p_{\text{cls}}^{t} \|_2], \qquad \mathcal{L}_{\text{reg}}^{\text{KD}} = \mathcal{L}_{\text{reg}}(p_{\text{reg}}^{s},\ p_{\text{reg}}^{t}), \tag{2}$$

where superscripts s and t indicate student and teacher, p_{cls} and p_{reg} represent the classification response after the sigmoid and bounding box regression prediction of detector separately, κ is the bilinear interpolation to match student output resolutions towards teacher, $\mathcal{L}_{\text{reg}}$ is the regression loss function of 3D detector and m_{cls} is a mask ranged in $[0, 1]$ to indicate important regions in p_{cls} (see green parts in Figure 3). Compared to vanilla KD [18] which mimics all teacher outputs, GID-L only focuses on some local regions covered by selected box proposals and results in different m_{cls}.

Feature KD is the major stream of work in KD for 2D object detection. It enforces student models to mimic teacher models' intermediate feature maps. Specifically, we construct feature mimicking on the last layer of BFE between student and teacher network as follows:

$$\mathcal{L}_{\text{feat}}^{\text{KD}} = \mathbb{E}[m_{\text{feat}} \| \psi(\phi(\kappa(f^{s})), y) - \psi(f^{t}, y) \|_2], \tag{3}$$

where y is the ground truth, ψ indicates the RoI Align [15], ϕ is a 1×1 convolution with batch normalization [21] and ReLU [34] block to align channel-wise discrepancy between teacher feature f^t and student feature f^s, m_{feat} is the mask to indicate critical regions ranged in $[0, 1]$ (see purple parts in Figure 3). Considering the imbalance between foreground and background regions in the detection, a prevailing solution is to emphasize near object regions to perform distillation [26, 50, 9]. Different investigated feature KD methods mainly vary in the selection of such critical regions m_{feat}.

Label KD is a newly proposed distillation strategy, which leverages teacher predictions in the label assignment stage of student [35]. Motivated by the simple and general label KD, we also employ it as a KD baseline method. Specifically, given a point cloud scene x and its corresponding ground truth (GT) set y, label KD first obtains the prediction y^t and confidence score o^t from the pretrained teacher detector. After filtering o^t with a given threshold τ, it then obtains a high-quality teacher prediction set $\hat{y}^t$. It generates a teacher assisted GT set $\hat{y}^{\text{KD}} = \{y, \hat{y}^t\}$ by combining GTs as well as confident teacher predictions and carries on label assignment for student with $\hat{y}^{\text{KD}}$. The final classification and regression losses with label KD on student detectors are $\hat{\mathcal{L}}_{\text{cls}}$ and $\hat{\mathcal{L}}_{\text{reg}}$.

Training Objective is the combination of three stream KD techniques as follows:

$$\mathcal{L} = \hat{\mathcal{L}}_{\text{cls}} + \lambda \hat{\mathcal{L}}_{\text{reg}} + \alpha_1 \mathcal{L}_{\text{cls}}^{\text{KD}} + \alpha_2 \mathcal{L}_{\text{reg}}^{\text{KD}} + \alpha_3 \mathcal{L}_{\text{feat}}^{\text{KD}}, \tag{4}$$

where λ, α_1, α_2 and α_3 are trade-off parameters between different objectives. Note that the existence and implementation of each operation (*e.g.* ψ, κ, m_{feat}, m_{cls}, etc) varies among different KD methods.

Table 3: Knowledge distillation benchmark for 3D detection on Waymo. Performance are measured in LEVEL 2 mAPH. Best and second-best methods are noted by **bold** and underline, respectively. "Ours" indicates our proposed improved knowledge distillation method introduced in Sec. 5

Detector	No Distill	Logit KD		Feature KD				Label KD	Ours	Flops (G)	Acts (M)
		KD	GID-L	FitNet	Mimic	FG	GID-F				
CP-Pillar	59.09	-	-	-	-	-	-	-	-	333.9	303.0
CP-Pillar-v0.4	57.55	57.51	57.54	57.89	58.57	58.44	58.26	58.10	**59.24**	212.9	197.7
CP-Pillar-v0.48	56.27	55.76	56.29	55.82	57.26	57.26	57.23	57.54	**58.53**	149.4	142.3
CP-Pillar-v0.64	52.81	53.13	50.78	51.79	53.83	53.37	53.18	53.78	**55.82**	85.1	88.0
CP-Voxel	64.29	-	-	-	-	-	-	-	-	114.7	101.9
CP-Voxel-S	62.23	62.81	62.89	60.51	63.35	63.33	62.75	63.31	**64.25**	47.8	65.7
CP-Voxel-XS	61.16	61.30	62.25	58.94	62.23	62.48	62.42	61.81	**63.53**	36.9	58.4
CP-Voxel-XXS	56.26	56.11	57.19	52.24	57.00	57.92	57.16	57.02	**59.28**	12.0	33.1

4.2 Results and Analysis

Benchmark Analysis. As shown in Table 3, compared to the no distillation baseline, all three streams of KD methods obtain performance improvements on six teacher-student pairs. Among seven KD baseline strategies, feature-based KD methods (*i.e.* Mimic and FG) achieve prominent performance, which demonstrates the strong potential of learning from teacher's hints on feature

extraction. Furthermore, we find that instance-aware local region imitation is important in distillation for 3D detection, as enormous background regions overwhelm the supervision of sparse instances. For instance, with instance-aware imitation, Mimic, FG and GID-F consistently outperform FitNet which fully imitates all spatial positions of teacher feature maps. Similar conclusions can also be drawn in logit KD by comparing the results of instance-aware GID-L and vanilla KD.

Synergy Analysis. While benchmark results in Table 3 mainly focus on the individual effectiveness of each KD manner, their synergy effect is also an important consideration, which has the potential to further improve student performance. As shown in Table 4, although feature KD itself achieves the highest performance on CP-Voxel-XXS compared to logit KD and label KD techniques, it can hardly achieve improvements or even suffers from performance degradation when combined with other KD methods. On the contrary, logit KD and label KD can cooperate well with each other to further improve student's capability. This is potentially caused by logit KD and label KD implicitly enforcing regularization on the feature, which can be conflicted with the optimization direction of feature KD and results in a poor synergy effect.

Table 4: Synergy investigation based on CP-Voxel-XXS.

GID-L	Label KD	FG	mAPH
			56.26
√			57.19
	√		57.02
		√	57.92
√	√		57.60
√		√	**58.01**
	√	√	57.06
√	√	√	57.62

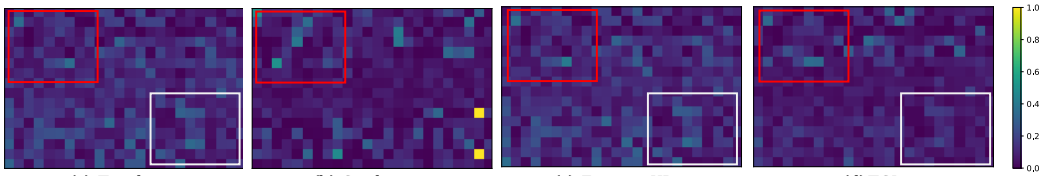
5 Improved Knowledge Distillation for 3D Object Detection

As shown in Table 4, the basic knowledge distillation pipeline fails to achieve remarkable results by combining the best method of three KD streams (*i.e.* GID-L [9], label KD [35] and FG [50]). In the following, we propose an improved KD pipeline for 3D object detection, including pivotal position logit KD to alleviate the extreme imbalance between foreground and background regions through only imitating response on sparse pivotal positions, label KD, and teacher guided initialization scheme to further facilitate transferring teacher's feature extraction ability to student models.

5.1 Method

Pivotal Position Logit KD. Motivated by the imbalance of foreground and background regions, previous 2D methods attempt to only enforce output-level imitation on pixels near or covering instances [50, 9]. However, we find that it is sub-optimal in 3D scenarios given more extreme imbalance between small informative instances and large redundant background areas. For example, based on CP-Pillar, even a vehicle with $10m$ length and $4m$ width occupies only 32×13 pixels in the final 468×468 response map. Such small instances and large perception ranges in 3D detection requires more sophisticated imitation region selection than previous coarse instance-wise masking manners in 2D detection. Hence, we propose Pivotal Position (PP) logit KD which leverages cues in teacher classification response or label assignment to determine the important areas for distillation.

Specifically, pivotal position selection can be formulated as finding suitable m_{cls} in Eq. (2). Here, we show three variants to obtain it. First, confidence of high-performance teacher prediction can serve as a valuable indicator to figure out pivotal positions for student (*i.e.* confidence PP). By filtering teacher confidence o^t with a threshold τ_{pp}, we then set i, j positions of m_{cls} with $o^t_{i,j} \geq \tau_{pp}$ to one and otherwise zero. With similar spirit, we can also select top-ranked K positions (*i.e.* rank PP) in teacher classification response P^t_{cls} as pivotal positions and convert them to a one-hot embedded m_{cls}. These confident or top-ranked positions are shown to be near object centers or error-prone regions. Last, inspired by the Gaussian label assignment in CenterPoint [58], we can define pivotal positions in a soft way with center-peak Gaussian distribution for each instance as m_{cls} (*i.e.* Gaussian PP). We empirically show that all three variants of PP logit KD achieve promising gains in 3D detection.



(a) Teacher (b) Student (c) Feature KD (d)TGI

Figure 4: Visualization of channel-wise L_1 norm for distillation from CP-Pillar to CP-Pillar-v0.64.

Teacher Guided Initialization. As mentioned in Section 4.2, feature KD itself outperforms other KD methods, but its deficient synergy results hinder 3D detection KD pipeline from achieving promising performance. This might be caused by the conflict between different optimization directions on

intermediate features from five KD loss terms in Eq. (4). Hence, we explore whether there is a substitute manner which can also leverage teacher's guidance on the feature extraction aspect. Since the value of feature map is determined by model's weights, finding hints from teacher weights becomes an alternative to guide student's feature extraction.

Hence, we propose to directly use the trained weights of teacher to serve as the initialization of student network, named teacher guided initialization (**TGI**) to enhance student model's feature extraction abilities by inheriting it from a teacher model. Although such initialization is intuitive for input compressed students (*e.g.* CP-Pillar-v0.64), it cannot be directly applied to pruned students (*e.g.* CP-Voxel-S). Therefore, we employ the parameter remapping strategy FNA [11] to project teacher weights on student parameters. Take 2D convolution layer as an example, teacher and student parameters are represented with $W^t \in \mathbb{R}^{r\times q\times h\times w}$ and $W^s \in \mathbb{R}^{v\times u\times h\times w}$, where $v \leq r$ and $u \leq q$, respectively. The first v and u channels of teacher parameters are directly assigned to the slimmed student network in FNA. We visualize the channel-wise L_1 norm of backbone feature to compare TGI extracted features with other strategies. As shown in Figure 4, our TGI extracts feature similar to teacher's (see red box region), but not naively mimics all teacher channels as feature KD does (see white box region). We argue that fully imitating all teacher features might penalize student optimization regarding the architecture or input resolution discrepancy between teacher-student pairs. We empirically show that our simple TGI strategy achieves comparable or even better performance individually among four feature KD methods and shows prominent synergy effects on six teacher-student pairs (See Sec. C).

5.2 Main Results and Comparison

To verify our improved KD pipeline, we conduct experiments on all six teacher-student pairs. As shown in Table 3, our improved KD pipeline consistently surpasses previous KD strategies on all settings with $0.7\% \sim 1.9\%$ improvement, thanks to our enhanced logit KD and more collaborative TGI. Equipping our lightweight detectors with the improved KD pipeline, CP-Voxel-S obtains comparable performance to CP-Voxel in terms of mAPH with around $2.4\times$ fewer flops and $1.6\times$ fewer activations. Furthermore, with $3.1\times$ fewer flops, CP-Voxel-XS only suffers 0.76% performance degradation. In addition, with around $1.6\times$ fewer flops and $1.5\times$ fewer activations, our distilled CP-Pillar-v0.4 even slightly outperforms CP-Pillar. CP-Pillar-v0.64 requires only 25% flops and 29% activations of teacher model, while achieving 55.82% mAPH, only 3.27% performance drop compared to CP-Pillar. These experimental results demonstrate that knowledge distillation is a promising technique to improve the performance of efficient 3D detectors.

5.3 Comparison with Other Detectors

To further demonstrate the efficiency and effectiveness of our designed detectors and KD pipeline, we also compare our distilled students with other detectors on the full WOD. As shown in Table 5, our CP-Voxel-S even outperforms its teacher CP-Voxel with around $2.0\times$ fewer parameters, $2.4\times$ fewer flops and $1.6\times$ activations. With similar latency and much fewer parameters, flops as well as activations, CP-Voxel-XS outperforms SECOND by 7.6%. Our CP-Pillar-v0.64, is $2.4\times$ faster than PointPillar on a GTX-1060 while achieves 1.8% higher performance.

Table 5: Comparison with other detectors on full WOD. † indicates results re-implemented by us.

Detector	Params (M)	Flops (G)	Acts (M)	Latency (ms)	LEVEL 2 mAPH
PointPillar† [24]	4.8	255.0	233.5	129.1	57.05
SECOND† [55]	5.3	84.5	76.4	84.6	57.23
CP-Pillar† [58]	5.2	333.9	303.0	157.9	61.56
CP-Voxel† [58]	7.8	114.8	101.9	125.7	65.58
PV-RCNN [41]	13.1	117.7	399.4	623.2	63.33
PV-RCNN++† [42]	16.1	123.5	179.7	435.9	**69.46**
CP-Voxel-S + Ours	4.0	47.8	65.7	98.0	**65.75**
CP-Voxel-XS + Ours	2.8	36.9	58.4	88.1	64.83
CP-Voxel-XXS + Ours	1.0	12.0	33.1	70.4	60.93
CP-Pillar-v0.4 + Ours	5.2	212.9	197.7	103.4	61.60
CP-Pillar-v0.48 + Ours	5.2	149.4	142.3	81.9	60.95
CP-Pillar-v0.64 + Ours	5.2	85.1	88.0	54.5	58.89

5.4 Cross Stage Distillation

A prevailing strategy to improve state-of-the-art 3D object detectors is to adopt a object proposal refinement head for two stage detection [41, 10, 42]. However, despite performance improvements by 3.5%, PVRCNN++ requires around $2.2\times$ parameters, $1.8\times$ activations and $3.5\times$ latency compared to CP-Voxel (see Table 6). Such computation and parameter overheads hinder the real-world applications of state-of-the-art two-stage 3D detectors. Therefore, here we also investigate whether the knowledge of the two-stage detector can help the learning of single-stage detector. It is noteworthy that this is the first attempt at

Table 6: Cross stage 3D detector distillation.

Detector	Params (M)	Flops (G)	Acts (M)	Latency (ms)	Mem. (G)	LEVEL 2 mAPH
PV-RCNN++	16.1	123.5	179.7	435.9	4.2	67.80
CP-Voxel	7.8	114.8	101.9	125.7	2.8	64.29
CP-Voxel + Ours	7.8	114.8	101.9	125.7	2.8	**65.27**

cross stage distillation in both 2D and 3D object detection. As shown in Table 6, leveraging hints from pretrained PVRCNN++, our distilled CP-Voxel achieves around 1% performance gains without any extra computation and parameter overheads during inference.

5.5 Generalization to More Scenarios

To demonstrate the generality of our compression and knowledge distillation manners, we provide experiments on other dataset, detector and compression manners. Besides, we construct experiment and discussion on extending our method to other task and more detectors in the Sec. E and F.

Generality on Other Dataset and Detector. As shown in Table 7 and Table 8, both our compression conclusion in Sec. 3.3 and the improved KD method can generalize well to KITTI [12] dataset with a new anchor-based detector SECOND. Especially, the SECOND (a) surpasses teacher performance by around 0.5% with 3.5× fewer flops, showing the generality of our conclusion and methods.

Table 7: Model and input compression results on KITTI. Teacher models are marked in gray.

Detector		Architecture: Width PFE	Architecture: Width BFE	Architecture: Voxel Size (m)	Efficiency: Params (M)	Efficiency: Flops (G)	Efficiency: Acts (M)	Efficiency: Latency (ms)	Moderate mAP@R40	CPR
SECOND		1.00	1.00	0.05	5.3	80.5	69.3	77.4	67.24	-
	(a)	0.75	0.50	0.05	1.6	23.0	38.0	51.8	65.62	0.69
	(b)	0.50	0.50	0.05	1.4	20.5	35.9	46.1	64.21	0.68
	(c)	0.50	1.00	0.05	4.6	72.4	65.2	70.6	65.70	0.50
	(d)	1.00	1.00	0.10	5.3	21.2	19.4	34.2	54.32	0.62

Table 8: Knowledge distillation results for 3D detection on KITTI. Performance are measured in moderate mAP over 40 recall positions. Best method is noted by **bold**.

Detector	No Distill	Logit KD: KD	Logit KD: GID-L	Feature KD: FitNet	Feature KD: Mimic	Feature KD: FG	Feature KD: GID-F	Label KD	Ours	Flops (G)	Acts (M)
SECOND	67.24	-	-	-	-	-	-	-	-	80.5	69.3
SECOND (a)	65.62	66.06	66.34	66.00	66.37	66.58	66.75	67.03	**67.70**	23.0	38.0

Generality on Advance Compression Manner. As shown in Table 9, since the coarser-resolution detector has more architecture-level redundancy with less input information, CP-Pillar (f) achieves higher CPR by combining the model and input compression strategies. Besides, after applying KD methods to CP-Pillar (f) as Table 10, it is still more efficient though fewer improvements from KD strategies with less redundancy. This demonstrates that our pipeline can substantially obtain a more efficient detector by harvesting the progress of advanced compression and distillation manners.

Table 9: Integrating different compression manners on Waymo. Teacher model is marked in gray.

Detector	Architecture: Width PFE	Architecture: Width BFE	Architecture: Width Head	Architecture: Voxel Size (m)	Efficiency: Params (M)	Efficiency: Flops (G)	Efficiency: Acts (M)	Efficiency: Latency (ms)	LEVEL 2 mAPH	CPR
CP-Pillar	1.00	1.00	1.00	0.32	5.2	333.9	303.0	157.9	59.09	-
CP-Pillar-v0.4	1.00	1.00	1.00	0.40	5.2	212.9	197.7	103.4	57.55	0.64
CP-Pillar (e)	1.00	0.875	0.875	0.32	4.0	260.1	267.7	134.7	58.53	0.54
CP-Pillar (f)	1.00	0.875	0.875	0.40	4.0	163.9	175.5	92.1	57.36	0.67

6 Ablation Studies

In this section, we conduct extensive ablation experiments to in-depth investigate the effectiveness of each component in our improved KD pipeline.

Component Ablation. Here, we investigate each component of our KD method and their synergy results. As shown in Table 11, based on CP-Voxel-XXS, both PP logit KD and TGI can obtain around 1.4% improvements separately. When incorporating with each other and label KD, they further obtain around 1.6% gains, showing the prominent synergy impact of our components. On the contrary, feature KD even suffers $0.2 \sim 0.6\%$ performance drop when combined with other KD techniques.

Investigation of TGI. We study how different parameter remapping manners and teachers influence the TGI. Besides FNA, we also attempt other parameter remapping strategies for TGI, including OFA [3] and Slim [32] by selecting important channels with designed indicators. As shown in Table 12, based on CP-Voxel-XS, OFA and Slim obtain inferior results compared to FNA, since indicator guided channel selection cannot determine consistent channel mapping for skip connection [16], while FNA survives by simply selecting beginning channels of all layers. In addition, CP-Voxel-XXS

Table 10: Knowledge distillation results for more sophisticated compressed 3D detector on Waymo. Teacher model is marked by gray Best method is noted by **bold**. § indicates the CPR is calculated according to the performance of best distilled student.

Detector	No Distill	Logit KD		Feature KD				Label KD	Ours	Flops (G)	Acts (M)	CPR§
		KD	GID-L	FitNet	Mimic	FG	GID-F					
CP-Pillar	59.09	-	-	-	-	-	-	-	-	333.9	303.0	-
CP-Pillar-v0.4	57.55	57.51	57.54	57.89	58.57	58.44	58.26	58.10	**59.24**	212.9	197.7	0.68
CP-Pillar (f)	57.36	56.93	56.70	57.15	57.81	57.48	57.77	57.57	**58.62**	163.9	175.5	0.70

Table 11: Component ablation study based on CP-Voxel-XXS.

PP logit KD	Label KD	TGI	Feature KD	mAPH
				56.26
√				57.68
		√		57.61
√		√		58.83
√	√			58.49
√	√	√		**59.28**
√	√		√	58.28
√	√	√	√	58.67

Table 12: Different remap manners studies of TGI based on CP-Voxel-XS and CP-Voxel-XXS.

Student	Teacher	Remap	mAPH
CP-Voxel-XS	-	-	61.16
	CP-Voxel	FNA	62.43
	CP-Voxel	OFA	60.06
	CP-Voxel	Slim	61.74
CP-Voxel-XXS	-	-	56.26
	CP-Voxel	FNA	54.92
	CP-Voxel-XS	FNA	57.61

only improves by inheriting parameters from CP-Voxel-XS but not CP-Voxel, which indicates a large architecture discrepancy between teacher and student hinders the effectiveness of TGI.

Different Variants of PP Logit KD. Here, we analyze different variants of PP logit KD (*i.e.* pivotal position selection according to teacher confidence, response ranking or soft Gaussian instance mask) in Section 5.1 based on CP-Voxel-S. While previous coarse instance-aware response imitation method GID-L obtains around 0.7% improvements, all three PP logit KD variants achieve around $1.6\% \sim 1.9\%$ gains. This demonstrates the significance of focusing on only vital positions in output-level distillation for 3D detection.

Table 13: Analysis of different PP logit KD variants based on CP-Voxel-S.

Logit KD Manner	mAPH
None	62.23
GID-L [9]	62.89
Confidence PP	63.84
Rank PP	63.85
Gaussian PP	**64.16**

Table 14: Analysis of label KD based on CP-Pillar-v0.48.

	Ground Truth		Teacher Pred		mAPH
	Cls	Reg	Cls	Reg	
(a)	all	all	no	no	56.24
(b)	all	all	all	no	56.47
(c)	all	non-overlap	all	all	56.99
(d)	all	all	all	non-duplicate	56.74
(e)	all	all	all	all	**57.54**

Analysis of Label KD. Although label KD has been proposed by the recent work [35], it just attributed its improvement to the "dark knowledge" from teacher. Here, we attempt to analyze how teacher predictions help student based on CP-Pillar-v0.48, where label KD performs best. As shown in Table 14, only integrating teacher prediction for classification label assignment, (b) achieves minor gains, which indicates that label KD mainly benefits the regression objective of student. When removing GTs highly overlapped with teacher predictions, (c) still achieves around 0.8% gains, which demonstrates that more achievable regression targets from teacher prediction facilitate student optimization. Last, by removing teacher predictions with object centers in the same BEV grid (*i.e.* duplicate objectives for single position), we find that (d) suffers performance drop compared to (e), which shows that duplicate regression objectives can also boost student regression ability.

7 Conclusion

We have examined the potential of knowledge distillation to serve as a generic method for obtaining efficient 3D detectors with extensive experimental results and analysis. We found that pillar-based detector prefers input compression while voxel-based detector is more suitable for width compression in designing efficient student models. Besides, we proposed pivotal position logit KD and teacher guided initialization for enhancing the 3D KD pipeline. Our best performing detector outperforms its teacher with $2.4\times$ fewer flops and our most efficient detector is $2.2\times$ faster than previous fastest voxel/pillar-based detector PointPillars on an NVIDIA A100 with higher performance. We hope our benchmark and analysis could inspire future investigations on this problem.

Acknowledgement. This work has been supported by Hong Kong Research Grant Council - Early Career Scheme (Grant No. 27209621), HKU Startup Fund, and HKU Seed Fund for Basic Research.

References

[1] Alex Bewley, Pei Sun, Thomas Mensink, Dragomir Anguelov, and Cristian Sminchisescu. Range conditioned dilated convolutions for scale invariant 3d object detection. *arXiv preprint arXiv:2005.09927*, 2020. 1, 2, 22

[2] Holger Caesar, Varun Bankiti, Alex H 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 *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11621–11631, 2020. 1

[3] Han Cai, Chuang Gan, Tianzhe Wang, Zhekai Zhang, and Song Han. Once-for-all: Train one network and specialize it for efficient deployment. *arXiv preprint arXiv:1908.09791*, 2019. 9

[4] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *European conference on computer vision*, pages 213–229. Springer, 2020. 21

[5] Defang Chen, Jian-Ping Mei, Yuan Zhang, Can Wang, Zhe Wang, Yan Feng, and Chun Chen. Cross-layer distillation with semantic calibration. In *AAAI*, 2021. 3

[6] Yilun Chen, Shu Liu, Xiaoyong Shen, and Jiaya Jia. Fast point r-cnn. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9775–9784, 2019. 1, 2, 3, 22

[7] Christopher Choy, JunYoung Gwak, and Silvio Savarese. 4d spatio-temporal convnets: Minkowski convolutional neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3075–3084, 2019. 21

[8] Angela Dai, Angel X Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5828–5839, 2017. 21

[9] Xing Dai, Zeren Jiang, Zhao Wu, Yiping Bao, Zhicheng Wang, Si Liu, and Erjin Zhou. General instance distillation for object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7842–7851, 2021. 1, 2, 3, 5, 6, 7, 10, 16, 17

[10] Jiajun Deng, Shaoshuai Shi, Peiwei Li, Wengang Zhou, Yanyong Zhang, and Houqiang Li. Voxel r-cnn: Towards high performance voxel-based 3d object detection. *arXiv preprint arXiv:2012.15712*, 1(2):4, 2020. 1, 3, 8, 19

[11] Jiemin Fang, Yuzhu Sun, Kangjian Peng, Qian Zhang, Yuan Li, Wenyu Liu, and Xinggang Wang. Fast neural network adaptation via parameter remapping and architecture search. *arXiv preprint arXiv:2001.02525*, 2020. 8

[12] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *2012 IEEE conference on computer vision and pattern recognition*, pages 3354–3361. IEEE, 2012. 1, 9

[13] Benjamin Graham and Laurens van der Maaten. Submanifold sparse convolutional networks. *arXiv preprint arXiv:1706.01307*, 2017. 1, 4, 19

[14] Xiaoyang Guo, Shaoshuai Shi, Xiaogang Wang, and Hongsheng Li. Liga-stereo: Learning lidar geometry aware representations for stereo-based 3d detector. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3153–3163, 2021. 3

[15] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017. 6

[16] 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. 2, 9

[17] Byeongho Heo, Minsik Lee, Sangdoo Yun, and Jin Young Choi. Knowledge transfer via distillation of activation boundaries formed by hidden neurons. In *AAAI*, 2019. 3

[18] Geoffrey Hinton, Oriol Vinyals, Jeff Dean, et al. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2(7), 2015. 1, 2, 3, 5, 6, 16, 18, 21

[19] Yuenan Hou, Zheng Ma, Chunxiao Liu, and Chen Change Loy. Learning lightweight lane detection cnns by self attention distillation. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 1013–1021, 2019. 1

[20] Zehao Huang and Naiyan Wang. Like what you like: Knowledge distill via neuron selectivity transfer. *arXiv:1707.01219*, 2017. 3

[21] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. PMLR, 2015. 6

[22] Xiao Jin, Baoyun Peng, Yichao Wu, Yu Liu, Jiaheng Liu, Ding Liang, Junjie Yan, and Xiaolin Hu. Knowledge distillation via route constrained optimization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019. 3

[23] Nikos Komodakis and Sergey Zagoruyko. Paying more attention to attention: improving the performance of convolutional neural networks via attention transfer. In *International Conference on Learning Representations*, 2017. 3

[24] Alex H Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. Pointpillars: Fast encoders for object detection from point clouds. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 12697–12705, 2019. 1, 2, 3, 8

[25] Bailin Li, Bowen Wu, Jiang Su, and Guangrun Wang. Eagleeye: Fast sub-net evaluation for efficient neural network pruning. In *European conference on computer vision*, pages 639–654. Springer, 2020. 22

[26] Quanquan Li, Shengying Jin, and Junjie Yan. Mimicking very efficient network for object detection. In *Proceedings of the ieee conference on computer vision and pattern recognition*, pages 6356–6364, 2017. 2, 3, 5, 6, 16, 17

[27] Yanghao Li, Saining Xie, Xinlei Chen, Piotr Dollar, Kaiming He, and Ross Girshick. Benchmarking detection transfer learning with vision transformers. *arXiv preprint arXiv:2111.11429*, 2021. 3

[28] Zhidong Liang, Ming Zhang, Zehan Zhang, Xian Zhao, and Shiliang Pu. Rangercnn: Towards fast and accurate 3d object detection with range image representation. *arXiv preprint arXiv:2009.00206*, 2020. 22

[29] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017. 18

[30] Yifan Liu, Ke Chen, Chris Liu, Zengchang Qin, Zhenbo Luo, and Jingdong Wang. Structured knowledge distillation for semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2604–2613, 2019. 1

[31] Zhengzhe Liu, Xiaojuan Qi, and Chi-Wing Fu. 3d-to-2d distillation for indoor scene parsing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4464–4474, 2021. 3

[32] Zhuang Liu, Jianguo Li, Zhiqiang Shen, Gao Huang, Shoumeng Yan, and Changshui Zhang. Learning efficient convolutional networks through network slimming. In *Proceedings of the IEEE international conference on computer vision*, pages 2736–2744, 2017. 9, 22

[33] Gregory P Meyer, Ankit Laddha, Eric Kee, Carlos Vallespi-Gonzalez, and Carl K Wellington. Lasernet: An efficient probabilistic 3d object detector for autonomous driving. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12677–12686, 2019. 22

[34] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *International conference on machine learning*, 2010. 6

[35] Chuong H Nguyen, Thuy C Nguyen, Tuan N Tang, and Nam LH Phan. Improving object detection by label assignment distillation. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 1005–1014, 2022. 2, 3, 5, 6, 7, 10

[36] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in Neural Information Processing Systems*, pages 5099–5108, 2017. 1, 2, 4, 22

[37] Lu Qi, Jason Kuen, Jiuxiang Gu, Zhe Lin, Yi Wang, Yukang Chen, Yanwei Li, and Jiaya Jia. Multi-scale aligned distillation for low-resolution detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14443–14453, 2021. 3, 4

[38] Ilija Radosavovic, Raj Prateek Kosaraju, Ross Girshick, Kaiming He, and Piotr Dollár. Designing network design spaces. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10428–10436, 2020. 3

[39] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28, 2015. 16

[40] Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. Fitnets: Hints for thin deep nets. *arXiv preprint arXiv:1412.6550*, 2014. 2, 3, 5, 16

[41] Shaoshuai Shi, Chaoxu Guo, Li Jiang, Zhe Wang, Jianping Shi, Xiaogang Wang, and Hongsheng Li. Pv-rcnn: Point-voxel feature set abstraction for 3d object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10529–10538, 2020. 1, 2, 3, 8

[42] Shaoshuai Shi, Li Jiang, Jiajun Deng, Zhe Wang, Chaoxu Guo, Jianping Shi, Xiaogang Wang, and Hongsheng Li. Pv-rcnn++: Point-voxel feature set abstraction with local vector representation for 3d object detection. *arXiv preprint arXiv:2102.00463*, 2021. 3, 8, 21

[43] Shaoshuai Shi, Xiaogang Wang, and Hongsheng Li. Pointrcnn: 3d object proposal generation and detection from point cloud. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 770–779, 2019. 1, 2, 22

[44] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 2

[45] 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, 3, 17

[46] Pei Sun, Weiyue Wang, Yuning Chai, Gamaleldin Elsayed, Alex Bewley, Xiao Zhang, Cristian Sminchisescu, and Dragomir Anguelov. Rsn: Range sparse net for efficient, accurate lidar 3d object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5725–5734, 2021. 2, 22

[47] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015. 2

[48] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International conference on machine learning*, pages 6105–6114. PMLR, 2019. 4

[49] OpenPCDet Development Team. Openpcdet: An open-source toolbox for 3d object detection from point clouds. https://github.com/open-mmlab/OpenPCDet, 2020. 3, 5

[50] Tao Wang, Li Yuan, Xiaopeng Zhang, and Jiashi Feng. Distilling object detectors with fine-grained feature imitation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4933–4942, 2019. 2, 3, 5, 6, 7, 16, 17

[51] Yue Wang, Alireza Fathi, Abhijit Kundu, David A Ross, Caroline Pantofaru, Tom Funkhouser, and Justin Solomon. Pillar-based object detection for autonomous driving. In *European Conference on Computer Vision*, pages 18–34. Springer, 2020. 1, 2, 3

[52] Yue Wang, Alireza Fathi, Jiajun Wu, Thomas Funkhouser, and Justin Solomon. Multi-frame to single-frame: Knowledge distillation for 3d object detection. *arXiv preprint arXiv:2009.11859*, 2020. 3

[53] Yue Wang and Justin M Solomon. Object dgcnn: 3d object detection using dynamic graphs. *Advances in Neural Information Processing Systems*, 34:20745–20758, 2021. 21

[54] Xiaohan Xing, Yuenan Hou, Hang Li, Yixuan Yuan, Hongsheng Li, and Max Q-H Meng. Categorical relation-preserving contrastive knowledge distillation for medical image classification. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 163–173. Springer, 2021. 1

[55] Yan Yan, Yuxing Mao, and Bo Li. Second: Sparsely embedded convolutional detection. *Sensors*, 18(10):3337, 2018. 1, 2, 3, 8, 21

[56] Zetong Yang, Yanan Sun, Shu Liu, Xiaoyong Shen, and Jiaya Jia. Std: Sparse-to-dense 3d object detector for point cloud. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1951–1960, 2019. 2, 22

[57] Zhendong Yang, Zhe Li, Xiaohu Jiang, Yuan Gong, Zehuan Yuan, Danpei Zhao, and Chun Yuan. Focal and global knowledge distillation for detectors. *arXiv preprint arXiv:2111.11837*, 2021. 3

[58] Tianwei Yin, Xingyi Zhou, and Philipp Krahenbuhl. Center-based 3d object detection and tracking. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11784–11793, 2021. 1, 3, 5, 7, 8, 19, 21

[59] Yifan Zhang, Qingyong Hu, Guoquan Xu, Yanxin Ma, Jianwei Wan, and Yulan Guo. Not all points are equal: Learning highly efficient point-based detectors for 3d lidar point clouds. *arXiv preprint arXiv:2203.11139*, 2022. 1, 2, 3, 19, 22

[60] Yin Zhou and Oncel Tuzel. Voxelnet: End-to-end learning for point cloud based 3d object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4490–4499, 2018. 1, 2, 3

[61] Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. Deformable detr: Deformable transformers for end-to-end object detection. In *International Conference on Learning Representations*, 2020. 21

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] We do not consider sophisticate layer-wise model compression strategies (see section 3). More details are described in supplemental file.
   (c) Did you discuss any potential negative societal impacts of your work? [N/A] Our work is only for academic research purpose.
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...

(a) Did you state the full set of assumptions of all theoretical results? [N/A]
(b) Did you include complete proofs of all theoretical results? [N/A]

3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] Our implementation details are posted in supplemental materials and code will be available.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] We follow the popular codebase OpenPCDet (see section 3).
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] It is posted in supplemental materials
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See supplemental materials. Most of experiments are trained with 8 NVIDIA 1080Ti. Few experiments are trained with 8 NVIDIA V100 or NVIDIA A100. Full set results on Waymo are trained with 16 NVIDIA 1080ti.

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] See our reference.
   (b) Did you mention the license of the assets? [No] They are public released for academic utilization.
   (c) Did you include any new assets either in the supplemental material or as a URL? [No]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [Yes] They are public released.
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [No] We only use public dataset.

5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

A Appendix

Outline

In this supplementary file, we provide more details and experiments not elaborated in our main paper due to page length limits:

- Sec. B: Implementation details and hyper-parameters of our 3D detection KD benchmark.
- Sec. C: Additional experimental results on synergy results of TGI, per-class results, error bar results, and focal loss attempts.
- Sec. D: More analysis, including latency comparisons on different accelerators, operation-level optimizations and detectors, and qualitative analysis of CPR.
- Sec. E: Generality of our method on 3D semantic segmentation.
- Sec. F: Discussion on other detectors such as sparse detection architectures and other input representations.
- Sec. G: Limitation analysis.

B Implementation Details for Our Benchmark

In this section, we describe the implementation of previous 2D KD methods in 3D object detection. Notice that most 2D detection KD methods are built on anchor-based detectors (*e.g.* Faster-RCNN [39]) and model compressed teacher-student pairs, so we modify them to adapt to anchor-free detectors and handle input resolution compression. On the other hand, we also provide detailed hyper-parameter values to help reproduce our results. Besides, we will also open-source our benchmark suit upon acceptance. Most of the experiments are trained with 8 NVIDIA 1080Ti, while a few experiments are trained with 8 NVIDIA V100 or 8 NVIDIA A100. Full set results on Waymo are trained with 16 NVIDIA 1080Ti or V100.

Logit KD. As for logit KD methods (*i.e.* vanilla KD [18] and GID-L [9]), vanilla KD use all ones m_{cls} to fully mimic all teacher outputs and set the mask m_{cls} to be all one. As for GID-L, the original anchor-wise region selection manner cannot be extended to input resolution compressed students, since the interpolation cannot handle the resolution mismatch of regression predictions p_{reg} from teacher and student models. In this regard, we refer to the ablation studies of the original paper and use ground truth boxes as critical region selection criteria. Specifically, we set the non-zero spatial positions in the assigned classification target heatmap to one in m_{cls}. The loss weight α_1 and α_2 of $\mathcal{L}_{\text{cls}}^{\text{KD}}$ and $\mathcal{L}_{\text{reg}}^{\text{KD}}$ are set to 15.0 and 0.2, respectively. Notice that the loss weight α_2 of regression term $\mathcal{L}_{\text{reg}}^{\text{KD}}$ in Eq. (4) will be set to 0 for input resolution compressed students.

As for our PP logit KD, the threshold of confidence PP is set to 0.3 by default and the rank K for rank PP is set to 500. Although the three variants of PP logit KD show similar performance, we use Gaussian PP by default since it does not need hyper-parameters adjustment when adopted by different teacher-student pairs.

Feature KD. For the implementation of different feature KD methods with Eq. (3), we only employ convolutional block ϕ for width compressed students to align the number of channels between f^s and f^t and only hire spatial interpolation κ for input compressed students. Besides, for methods that utilize RoI Align ψ to extract object-level features, we will not use the interpolation κ to avoid introducing extra interpolation errors. As for the implementation of each method, we directly align student and teacher full features without mask m_{feat} and RoI Align ψ in FitNet [40]. As for Mimic [26], we use the more sophisticated RoI Align ψ instead of the original spatial pyramid pooling to extract features for each GT, and construct imitation on the object-level features between teacher and student. As for FG [50], to extend its anchor-based critical region selection, we directly set the non-zero regions in the assigned classification target heatmap as the critical regions in m_{feat}, the other implementations are the same as FitNet. As for GID-F [9], we use teacher predictions after Non-Maximum Suppression (NMS) as critical regions and set their corresponding spatial positions in m_{feat} to one. Besides, we utilize RoI Align ψ to extract object-level features to calculate feature distillation loss and also apply the relation loss among different object features as the description in the original paper. The loss weight α_3 of feature KD loss $\mathcal{L}_{\text{feat}}^{\text{KD}}$ is set to 100 for input compressed students or 200 for width compressed students, respectively. The loss weight of the relation loss of GID-F is set to 0.1.

Label KD. There is only one work for label KD which has been described in the main paper. Notice that we do not hire NMS for teacher predictions for label assignment empirically. The score threshold τ to filter high-quality teacher predictions is set to 0.6 by default.

Among different 2D KD methods, we notice that FG [50], Mimic [26] and GID [9] highlight the critical region selection on feature KD or logit KD to tackle the imbalance between foreground and background regions in object detection. There are mainly two foreground-region imitation strategies: one is using RoI Align ψ to extract object-wise features with teacher prediction or GT boxes as guidance (*e.g.* Mimic [26] and GID-F [9]); the other is assigning a one-hot mask m_{feat} or m_{cls} to calculate imitation loss on some critical regions (*e.g.* FG [50] and GID-L [9]). We empirically find that RoI Align is more suitable for input compression setups since it avoids the interpolation errors when aligning spatial resolutions while the mask-based strategy is more flexible and general as it allows position-wise imitation. Comparing different critical region selection techniques, we empirically show that a key factor to make KD methods work well on 3D detection is to focus on only a few positions. For example, our proposed PP logit KD focus on only around $\frac{1}{5} \sim \frac{1}{20}$ positions of the positions selected by traditional 2D KD methods. This is caused by the fact that a spatial position in the 3D BEV features can represent a $0.8m \times 0.8m \times 6m$ pillar in the 3D geometry space, which is informative and can cover even a single pedestrian. In this regard, the critical region selection techniques are supposed to focus on fewer informative positions in the 3D detection setting.

C Additional Experimental Results

In this section, we provide some additional experimental results as a supplement to our main paper. This part consists of the full synergy results of TGI on six teacher-student pairs, per-class performance and error bar results.

C.1 Synergy Results of TGI

Table 15: Synergy results of TGI and feature KD on Waymo with six teacher-student pairs. Performance are measured in LEVEL 2 mAPH. Teacher results are masked by gray.

Detector	No Distill	Feature KD	Label KD	PP Logit KD	TGI	PP Logit KD + Feature KD	PP Logit KD + TGI	Label KD + Feature KD	Label KD + TGI
CP-Pillar	59.09	-	-	-	-	-	-	-	-
CP-Pillar-v0.4	57.55	58.57	58.10	58.21	59.03	58.18	59.24	58.35	59.19
CP-Pillar-v0.48	56.27	57.26	57.54	56.89	57.91	57.11	58.20	57.43	58.34
CP-Pillar-v0.64	52.81	53.83	53.78	54.32	54.30	54.14	55.55	54.24	55.59
CP-Voxel	64.29	-	-	-	-	-	-	-	-
CP-Voxel-S	62.23	63.35	63.31	64.16	63.48	63.58	64.18	62.62	63.50
CP-Voxel-XS	61.16	62.48	61.81	62.76	62.43	62.90	63.41	62.34	62.85
CP-Voxel-XXS	56.26	57.92	57.02	57.68	57.61	58.19	58.83	57.06	57.71

The poor synergy effect of feature KD is the main motivation for us to design TGI. Due to the page limitation, we only present its experimental results on CP-Voxel-XXS as an example. Here, we compare the synergy results of feature KD and TGI on six teacher-student pairs to further show the promising performance of our TGI. As shown in Table 15, the results obtained by combining TGI and label KD or PP logit KD consistently outperform the synergy results of feature KD on all 12 scenarios, manifesting that TGI collaborates better with other KD techniques. Furthermore, our TGI itself achieves comparable results or even surpasses feature KD among six teacher-student pairs. These experimental results strongly demonstrate that our proposed TGI can be a powerful substitute for feature KD to transfer the feature extraction ability from the teacher model.

C.2 Per-class Performance

We report the per-category performance of our efficient detectors on full Waymo Open Dataset [45] in Table 16. As illustrated in Table 16, comparing CP-Pillar-v0.64 and CP-Pillar, the performance gap mainly lies in pedestrians and cyclists (around 3% gap), while vehicle suffers around 1.5% gap. This might be caused by the fact that coarser input resolution penalizes the performance of small objects such as pedestrians and cyclists more severely. As for voxel-based detector CP-Voxel-XXS, we notice that its performance gap from teacher distributes more evenly on different categories

Table 16: Per-class performance on full Waymo dataset for our six distilled efficient student models. Performance are measured in mAP/mAPH. Teacher results are masked by gray. Best results are indicated by **bold**.

Detector	Vehicle		Pedestrian		Cyclist	
	LEVEL 1	LEVEL 2	LEVEL 1	LEVEL 2	LEVEL 1	LEVEL 2
CP-Pillar	72.75/72.24	64.48/64.02	74.01/64.06	65.74/56.76	**67.84/66.37**	**65.34/63.92**
CP-Pillar-v0.4 + Ours	**73.01/72.46**	**64.85/64.36**	**75.00/64.24**	**66.86/57.13**	67.18/65.76	64.69/63.32
CP-Pillar-v0.48 + Ours	72.42/71.85	64.42/63.89	74.38/63.74	66.26/56.62	66.19/64.76	63.72/62.34
CP-Pillar-v0.64 + Ours	71.37/70.77	63.30/62.75	71.45/61.05	63.22/53.86	63.87/62.39	61.48/60.05
CP-Voxel	**74.31/73.75**	**66.35/65.84**	76.19/70.10	68.44/62.82	71.76/70.63	69.16/68.07
CP-Voxel-S + Ours	74.28/73.72	66.17/65.66	**76.72/70.68**	**68.96/63.37**	**71.97/70.81**	**69.36/68.24**
CP-Voxel-XS + Ours	73.62/73.05	65.53/65.01	75.50/69.29	67.67/61.96	71.30/70.09	68.69/67.52
CP-Voxel-XXS + Ours	69.20/68.55	61.15/60.57	71.76/64.95	63.71/57.53	68.51/67.18	65.98/64.70

than CP-Pillar-v0.64, as the model width compression does not have special penalization on any categories.

C.3 Error Bar

Table 17: Repeat results of our different models on Waymo . We report the reproduced results with 5 rounds as well as their averaged results and standard variance. The performance is measured in LEVEL 2 mAPH.

Detector	Round 1	Round 2	Round 3	Round 4	Round 5	Average	Standard Variance
CP-Pillar	59.09	59.13	59.13	59.14	59.01	59.10	0.05
CP-Pillar-v0.64	52.81	52.75	52.85	52.85	53.25	52.90	0.20
CP-Pillar-v0.64 + Ours	55.75	55.82	56.02	55.75	55.73	55.81	0.12

Here, to show the robustness of our experimental results, we reproduce knowledge distillation on CP-Pillar-v0.64 five times and report the average and standard derivation of performance. As shown in Table 17, the performance of our distilled CP-Pillar-v0.64 is more stable than the student without distillation, which indicates that our improved KD pipeline can boost performance stably.

C.4 Focal Loss Results

As focal loss [29] is a widely-used solution for the foreground and background region imbalance issue, it is intuitive to also employ it as the distillation loss. In this regard, here we provide an experimental comparison between focal loss and our proposed PP logit KD for logit KD. As shown in Table 18, PP logit KD is around 0.7% and 8.2% higher than focal loss on CP-Voxel-XS and CP-Pillar-v0.64, respectively. As for CP-Pillar-v0.64, since the capability difference between teacher and student are large, focal loss even suffers performance degradation compared to vanilla KD, while our PP logit KD consistently brings performance boost. The reason for the inferior performance of focal loss for distillation is that it will emphasize regions that are most different among teacher and student pairs but not most information-rich areas. Those large prediction difference areas could be caused by the capability gap between teacher and student and thus renders focal loss a suboptimal strategy for student learning.

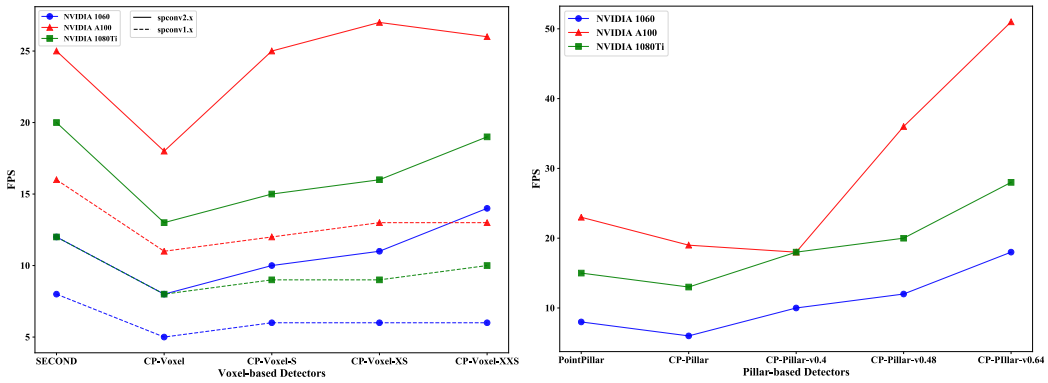
Table 18: Results of leveraging focal loss as logit distillation loss on Waymo. Teacher models are marked in gray.

Detector	No Distill	KD [18]	Focal loss	PP Logit KD
CP-Voxel	64.29	-	-	-
CP-Voxel-XS	62.23	62.81	63.48	64.16
CP-Pillar	59.09	-	-	-
CP-Pillar-v0.64	52.81	50.78	46.11	54.32

D More Analysis

In this section, we provide some investigations on the influence of accelerator types and operation-level optimizations on the measured latency as well as the qualitative analysis of our proposed CPR.

D.1 Latency Analysis



(a) FPS for different voxel-based detectors

(b) FPS for different pillar-based detectors

Figure 5: Comparison on the FPS with different hardware devices and operation-level optimizations for different detectors.

Table 19: Information of the inference machine.

Type	GPU	CPU
Personal Computer	NVIDIA GTX-1060	Intel(R) Core(TM) i7-7700K CPU @ 4.20GHz
Server#1	NVIDIA GTX-1080Ti	Intel(R) Xeon(R) CPU E5-2682 v4 @ 2.50GHz
Server#2	NVIDIA A100	AMD EPYC 7742 64-Core

Inference time, latency or FPS directly measure the execution speed of a model on a given hardware configuration, and have been widely adopted to assess the model efficiency in 3D detection. However, different papers [59, 58, 10] measure the latency based on different machines, hindering the fair comparisons and standardization among different approaches. Besides, we empirically show that the operation-level optimization has a large impact on the latency measurement and even influences the conclusion. In this regard, we investigate how different hardware devices and operation-level optimizations (*i.e.* sparse convolution [13]) affect the latency measurement in terms of FPS. The basic information of our tested machines is shown in Table 19, including one personal computer and two servers.

Influence of Hardware Devices. As shown in Figure 5 (a) and (b), detectors run faster on more powerful GPU consistently for both voxel-based and pillar-based detectors, which demonstrate the great influence of hardware devices. In addition, we notice that some compressed detectors meet negative optimization (*i.e.* model needing fewer computations has larger latency) on the latest GPU architecture NVIDIA-A100 (see CP-Voxel-XS *vs.* CP-Voxel-XXS and CP-Pillar *vs.* CP-Pillar-v0.4), which has not been observed on GTX-1060 and GTX-1080Ti. This might be caused by the different underlying implementation strategies at the hardware level. This also verifies that latency-orientated efficient detector designs largely depend on the type of hardware devices.

Operation-level Optimization. Sparse convolution network [13] is a major component in existing voxel-based detectors to efficiently extract voxel-wise features from voxelized point clouds. As they are not fully-optimized toward hardware, it occupies a large percentage of latency for voxel-based detectors using A popular implementation Spconv [1], although the GLOPs are not that high. Here, we also investigate how different implementations of the sparse convolution influence the measured latency. As shown in Figure 5 (a), voxel-based detectors implemented with Spconv2.x is much faster than their counterparts with Spconv1.x (Spconv2.x is the optimized version of Spconv1.x). Moreover, the operation-level optimization can even have a larger impact than the hardware devices (see NVIDIA 1080Ti with Spconv1.x and NVIDIA 1060 with Spconv2.x). In addition, although

[1] https://github.com/traveller59/spconv

the width-level compressed students of CP-Voxel require significantly fewer flops, parameters and activations compared to CP-Voxel, they cannot obtain obvious speed up on the latency with Spconv1.x, which indicates that non-parametric computations (*i.e.* computation not directly related to learnable parameters) occupy most of the latency for voxel-based detectors using Spconv1.x. As a consequence, based on NVIDIA 1060, CP-Voxel with Spconv2.x runs faster than CP-Pillar, while runs slower than CP-Pillar when using Spconv 1.x. This demonstrates that testing model latency on different operation-level optimizations can draw totally different conclusions when comparing the efficiency of different detectors in terms of latency.

Besides the above two factors, we also observe that even hardware status (*e.g.* temperature) could influence the final obtained latency, indicating that the latency is hard to stably reproduce on the same machine and cannot serve as a standard measurement on model efficiency. In this regard, we focus on the parametric measurement such as flops and activations in the main paper, as they will not be influenced by the above hardware, software or environment level factors.

D.2 Qualitative Analysis of CPR

Table 20: More model compression results. Teacher models are marked in gray. See text for details.

Detector		Architecture					Efficiency					LEVEL 2 mAPH	CPR
		Width			Depth		Params	Flops	Acts	Latency	Mem.		
		PFE	BFE	Head	PFE	BFE	(M)	(G)	(M)	(ms)	(G)		
CP-Pillar		1.00	1.00	1.00	1.00	1.00	5.2	333.9	303.0	157.9	5.2	59.09	-
	(a)	1.00	0.50	1.00	1.00	1.00	1.5	130.1	203.1	97.1	3.6	55.35	0.58
	(b)	1.00	0.25	0.25	1.00	1.00	0.3	23.8	91.2	51.9	2.3	46.16	0.59
	(c)	1.00	1.00	1.00	1.00	0.50	2.2	258.5	234.1	118.5	4.3	55.24	0.52
	(d)	1.00	1.00	1.00	1.00	0.33	1.4	234.6	210.0	107.8	4.0	47.97	0.42
CP-Voxel		1.00	1.00	1.00	1.00	1.00	7.8	114.7	101.9	125.7	2.8	64.29	-
	(a)	0.50	0.50	0.50	1.00	1.00	1.9	28.8	51.2	75.1	1.7	59.47	0.64
	(b)	0.50	0.25	0.25	1.00	1.00	1.0	12.0	33.1	70.4	1.3	56.26	0.67
	(c)	1.00	1.00	1.00	0.50	0.50	3.0	63.9	65.2	73.0	1.9	60.95	0.61
	(d)	1.00	1.00	1.00	0.33	0.33	1.8	47.9	52.2	59.0	1.6	55.78	0.57

When designing efficient student models, we propose CPR to quantitatively measure the trade offs between efficiency and performance of a compressed student model. Here, we take some examples to qualitatively analyze the correlation between the CPR, the model efficiency as well as the model accuracy. As shown in Table 20, comparing CP-Pillar (a) and (c), they achieve similar performance, but CP-Pillar (a) requires only half of flops and fewer parameters, activations, latency and training memory. This indicates that CP-Pillar (a) achieves better trade offs between accuracy and efficiency, which can be reflected on its higher CPR. Similar conclusions can be drawn by comparing CP-Pillar (b) and (d), CP-Voxel (a) and (c), as well as CP-Voxel (b) and (d). These qualitative results demonstrate the good correlation between CPR and the compromise between accuracy and efficiency for a given compressed model.

E Generality on 3D Semantic Segmentation

In this work, the above experiments are all built on 3D object detection, which is a sparse prediction task. However, we argue that our sparse distillation manner (*i.e.* pivotal position logit KD) can also generalize to dense prediction tasks such as 3D semantic segmentation. As the student model has dense GTs supervision in training, dense distillation loss on massive uninformative points and regions, such as road points, might be redundant and can overwhelm the overall distillation loss. Instead, our sparse distillation might help the student focus on more important areas by using teacher prediction as regularization.

Here, we follow the design principle of PP logit KD and adapt it to handle the dense semantic segmentation task. We apply distillation loss on points with predictions that are correct but less confident than the teacher. Our simple design is motivated by three intuitions: (i) Points that are correctly predicted with lower confidence are often some challenging cases that the model is struggling but also has the capability to handle. By harvesting knowledge from a high-performing teacher model, the student can learn to match the confidence level of the teacher which provides more information than the one-hot GT. (ii) Points that are correctly predicted with higher confidence are often easy samples that have very close prediction confidence to the teacher model. Considering that

these samples are already handled well by the model, they have low chance to benefit from distillation but might cause redundancies. (iii) Points that are incorrectly predicted by the student are often cases that might be out of the ability of student models. Specifically, we have the confidence of student predictions conf^s, the confidence of teacher predictions conf^t and a pre-defined threshold τ. We will only apply distillation loss for student predictions that are correct and have $\text{conf}^s + \tau < \text{conf}^t$.

We also provide experimental results for our design on the 3D semantic segmentation dataset ScanNet [8]. Here, we use a small version of MinkowskiNet [7] for fast verification. On the one hand, as shown in Table 21, we try both model width and input resolution compression to obtain efficient student models, and select MinkowskiNet14-v0.04 as the student model for KD due to its higher CPR. On the other hand, as shown in Table 22, we compare the effectiveness of KD [18], PP logit KD and TGI on MinkowskiNet14-v0.04, where both our proposed PP logit KD and TGI obtain improvements. In particular, our sparse PP logit KD surpasses the dense logit KD method with around 0.8% gains. Our statistics also show that our PP logit KD only leverages 19.03% points for distillation at the first epoch and 3.66% points for distillation at the last epoch. These experiments and statistics demonstrate that sparse distillation can also work on the dense prediction task.

Table 21: Model width and input resolution compression results of MinkowskiNet14 on ScanNet. The teacher model is marked in gray.

Architecture			Efficiency			mIoU	CPR
Model	Width	Voxel Size (m)	Params (M)	Flops (T)	Acts (M)		
MinkowskiNet14	1.0	0.02	1.7	46.2	27.9	65.77	-
MinkowskiNet14-w0.5	0.5	0.02	0.5	18.2	17.4	61.84	0.60
MinkowskiNet14-v0.04	1.0	0.04	1.7	5.7	8.9	62.82	0.78

Table 22: Knowledge distillation results of compressed MinkowskiNet14 on ScanNet.

Model	Role	No Distill	KD [18]	PP Logit KD	TGI	Flops (T)	Acts (M)
MinkowskiNet14	Teacher	65.77	-	-	-	46.2	27.9
MinkowskiNet14-v0.04	Student	62.82	63.65	64.40	64.22	5.7	8.9

F Discussion on Other Detectors

In this work, we mainly focus on dense detectors (*e.g.* CenterPoint [58], PV-RCNN++ [42], SECOND [55]) with the most popular pillar/voxel input representations. Here, we construct some discussion for the knowledge distillation on the sparse detectors and other input representations to further demonstrate the generality of our distillation manners.

F.1 Discussion for Sparse Detectors

As the emergency of DETR [4], object detectors that directly produce sparse prediction without post-processing become a new popular detection paradigm. Although we only try our KD manner on dense detector in the main paper, we argue that our sparse distillation is still applicable for sparse transformer-based detector such as DETR [4], Deformable DETR [61], Object DGCNN [53], etc.

On the one hand, sparse transformer-based detectors that directly make instance predictions actually rely on learning to some sparser reference points and corresponding position features. For example, each object query in Deformable DETR [61] or Object DGCNN [53] is decoded into a reference point and neighboring points in order to focus only on those most informative positions. On the other hand, although sparse detectors can directly generate sparse instance predictions, our sparse distillation (*i.e.* pivotal position KD) focuses on sparser and more fine-grained position-level information (see Figure 6). In this regard, it should still be applicable to sparse models with some specific modifications.

Here, we take Object DGCNN [53] as an example and provide two possible sparse distillation designs.

(1) As the transformer encoder and decoder of Object DGCNN are similar to Deformable DETR, it can be simply extended to a two-stage variant as Deformable DETR. In the two-stage variant, the transformer encoder will regard each pixel as an object query and construct a dense scoring on it, where top-score positions are picked as reference points. This is similar to our designed rank PP KD which enforces the student to imitate the prediction of teacher top-rank positions. Therefore,

we can directly apply our sparse rank PP KD to those dense scoring predictions between teacher and student. Besides, we will also carry on feature imitation on those teacher top-ranked positions between teacher and student.

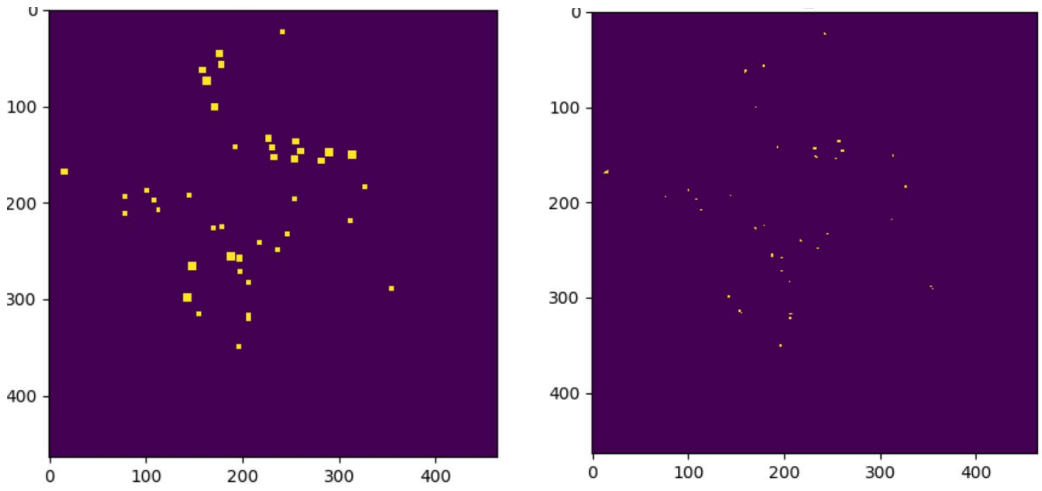


Figure 6: Visualization comparison of the imitation position (yellow positions) between instance-wise KD and our PP KD on the bird's eye view. Left: valid imitation regions for instance-wise KD. Right: valid imitation positions for our PP KD. Our PP KD has more fine-grained imitation regions compared to instance-wise KD. Best viewed in color.

(2) As for the one-stage variant of sparse detectors, learnable object queries will be decoded into reference points and neighboring points, so the sparse distillation can be constructed on those points and their corresponding BEV features. Specifically, we can first match the positive object queries of teacher and student as query pairs by checking whether they are matched to the same GT box. Then, we can enforce the decoded reference and neighboring points of the student to mimic their paired teacher counterparts. Besides, we will construct imitation on BEV features of those reference and neighboring positions between teacher and student.

F.2 Discussion for other input representations

Apart from the most popular pillar/voxel based object detectors discussed in the main paper, there are also point-based and range image based detectors. Therefore, we also provide the discussion on the point-based and range-based detectors here. As the TGI and label KD are detector-agnostic distillation manners and can be easily extended to detectors with any input representations, we only discuss the sparse distillation – pivotal position KD here.

Point-based detector [43, 6, 56, 59] take raw point clouds as input and and employed PointNet++ [36] to extract point features and generate point-wise object proposals. Range image based detectors leverage the native and dense representation for 3D points captured from LiDAR [33, 28, 1, 46]. As for the knowledge distillation on point-based and range image based detectors, since they still need to generate dense point-wise object proposals, our sparse distillation can still directly apply to it by selecting confident or top-ranked teacher positions for imitation (*i.e.* Confidence PP and Rank PP in Table 13). In this regard, our distillation strategies should be generalizable to all existing input representations.

G Limitations

Although our work has already investigated the compression on model width, model depth and input resolution for designing lightweight student detectors, there are also exhausted layer-wise model compression methods [32, 25], which have not been attempted in this work. Although missing the compression attempts in such perspective, we argue that our paper mainly focuses on exploring the potential of knowledge distillation to obtain efficient 3D detectors and the existing compression attempts already fulfill our demands. Besides, we believe that the further progress and investigation

of designing efficient 3D detectors is orthogonal to our KD attempts and can cooperate with our improved KD pipeline to obtain more efficient detectors.