ACCURATE 3D OBJECT DETECTION IN SUPER SPARSE SPACE VIA OPERATOR-LEVEL SPARSE ATTENTION

Anonymous authors
Paper under double-blind review

ABSTRACT

In autonomous driving perception systems, the extreme sparsity of LiDAR point clouds poses a fundamental challenge to achieving high-accuracy 3D object detection under constrained computational budgets. Prevailing paradigms adopt a "sparse to dense" pipeline, where the dense projection introduces significant computing and memory bottlenecks. Emerging fully sparse architectures eliminate this dense projection but suffer from severe inefficiencies in both computation and memory, stemming from indiscriminate processing of all voxels and the incompatibility of standard self-attention with irregular sparse sequences. Furthermore, conventional detection heads rely on fixed-range max pooling for peak localization, which fails under the fully sparse paradigm and degrades detection performance. To address these limitations, we propose an accurate fully sparse 3D detector. Specifically, a Sparse Transformer module based on Sparse Matrix Multiplication is designed to perform attention directly over non-empty voxels and their local window neighborhoods, completely eliminating redundant computation and memory overhead at empty locations. Furthermore, a Progressive Pruning Network is introduced to hierarchically prune low-confidence background voxels during feature extraction, progressively transforming the sparse voxel set into super sparse space that retains only foreground-enriched tokens, thereby enabling stepwise computational reduction as the network deepens. To tackle the peak localization issue, we present a Dynamic Transfusion Head that replaces fixed-range max pooling with learnable bounding-box-aware dynamic pooling. Evaluated on the nuScenes dataset, our method achieves new state-of-the-art accuracy and delivers an improved accuracy-efficiency trade-off, while requiring only half the training epochs of classic methods.

1 INTRODUCTION

Autonomous driving safety relies on long-range perception of critical road agents. 3D detectors must therefore provide high detection accuracy with resource-constrained on-board computing. Due to path dependence on established 2D pipelines, most existing 3D paradigms follow a sparse-to-dense pipeline [1, 2, 3, 4]. Given the high sparsity of point clouds, this conversion leaves extensive unoccupied BEV cells, expending substantial computation and memory on invalid locations. Moreover, both cost and memory scale quadratically with the perception range, making direct scaling to long-range, wide-field-of-view scenarios infeasible.

Against this backdrop, **fully sparse detection architectures** have become the mainstream for long-range efficient perception, as their cost scales approximately linearly with the number of valid points and is decoupled from the global perception range. The FSD series [5, 6] and VoxelNeXt [7] pioneered the paradigm of performing detection entirely on sparse voxels, eliminating the intermediate sparse-to-dense conversion step. Nevertheless, even with fully sparse architectures, a fundamental problem of computational waste remains. Since background voxels typically account for over 90% of non-empty voxels, and existing purely sparse detectors do not distinguish foreground from background, processing all voxels without screening consumes substantial computation on task-irrelevant background locations, causing considerable redundancy.

On the other hand, efficient voxel feature aggregation remains critical. Traditional sparse convolutions [1, 2, 7] aggregate locally through fixed receptive fields at low cost, yet are limited in modeling

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

long-range, content-dependent context. Self-attention has been incorporated into sparse voxel aggregation [8, 9, 10, 11] to improve modeling flexibility and accuracy. However, direct self-attention in sparse settings faces a fundamental efficiency barrier. Standard Transformers require fixed-length sequences, while sparse voxel sets are irregular and vary across frames. Existing methods therefore rely on zero-padding for sequence alignment, wasting computation and memory on padded positions.

Beyond efficiency, purely sparse architectures face an under-explored accuracy challenge: the absence of an explicit central peak search mechanism. Due to point-cloud sparsity, conventional max-pooling-based center selection is inapplicable, so existing approaches [5, 6, 7] resort to global top-k truncation followed by NMS. Without local comparison constraints, the top-k stage admits many duplicate candidates, increasing post-processing overhead; and mis-set thresholds may cause missed detections or redundant boxes, thereby affecting accuracy.

To address these challenges, we propose an **Accurate 3D Object Detection in Super Sparse Space via Operator-level Sparse Attention (DeS3)**. Specifically, a sparse Transformer (SpTransformer) module based on Sparse Matrix Multiplication (SPMM) is designed to compute self-attention directly on non-empty voxels within a window. To resolve the indiscriminate processing of foreground and background voxels, we integrate stage-wise foreground prediction heads into the backbone to prune background voxels progressively, transforming the sparse voxel set into a super sparse space. A Dynamic Transfusion Head further applies dynamic pooling within learnable bounding boxes, compensating for the missing peak-seeking mechanism. Our contributions are as follows:

- A method named DeS3 is proposed, in which the detection is restricted to an extremely sparse foreground space through progressive pruning.
- A window-level sparse Transformer backbone and a Dynamic Transfusion Head are devised for fully sparse architectures to systematically resolve the efficiency and accuracy bottlenecks.
- Systematic experiments conducted on the nuScenes [12] dataset demonstrate the superiority of our method in accuracy and memory usage.

2 Related Work

2.1 Feature Extraction: From Point-wise MLP to Transformer

Feature extraction for 3D detection evolves amid the accuracy–efficiency trade-off. The point-wise *MLP* paradigm [13, 14] preserves raw geometry but suffers limited local modeling and scalability. VoxelNet [15] addresses this via a voxel framework with VFE layers, 3D convolutions, BEV compression, and a 2D RPN. SECOND [1] replaces 3D convolutions with submanifold sparse convolution at non-empty voxels, now the de facto standard for voxel-based detection [2, 16, 17]. Yet sparse convolution is bounded by kernel size, limiting long-range modeling. To overcome convolution locality, self-attention was introduced into sparse voxel aggregation. SST [8] presents a sparse regional attention that partitions space into non-overlapping regions and employs region batching with bucket-based grouping. DSVT [9] delivers dynamic sparse window attention with rotated set partitioning along two scanning directions. Despite outperforming convolutional backbones, all these quasi-sparse Transformers follow a "grouping and zero-padding" pipeline to fit dense operators, leaving operator-level purely sparse attention open. In this work, we propose a genuine operator-level sparse Transformer based on window-level Sparse Matrix Multiplication (SPMM), which directly builds a sparse adjacency matrix between non-empty voxels and their window neighbors.

2.2 Dynamic Sparsification and Point Cloud Pruning

Since background voxels dominate the non-empty space, processing them without discrimination wastes computation on task-irrelevant regions. Dynamic sparsification aims to remove low-information voxels with minimal accuracy loss. SST [8] recognizes foreground sparsity and adopts single-stride feature extraction to avoid dilution, yet lacks explicit background removal. Ada3D [18] predicts BEV importance maps via a lightweight CNN, but its reliance on dense BEV projection is incompatible with purely sparse architectures. GSP [19] uses learnable Gumbel-Softmax pruning

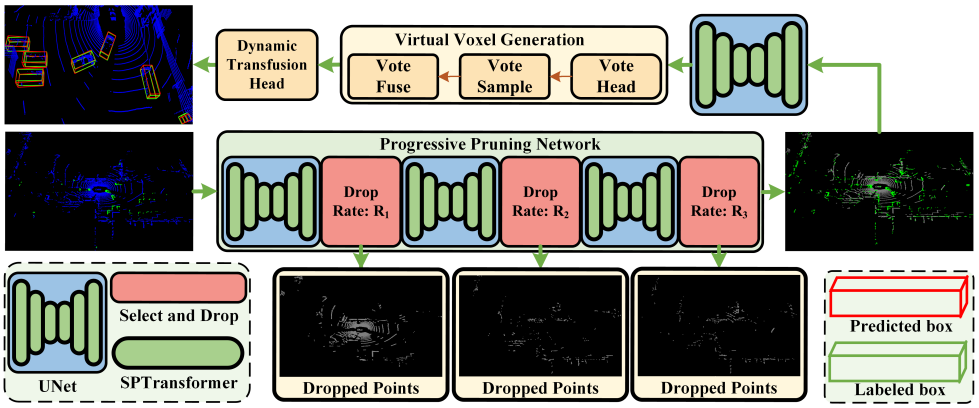

Figure 1: Overall framework of the proposed DeS3. The Select and Drop module will discard a part of voxels according to the predicted confidence level and the threshold value R.

layers for multi-frame accumulation, though its decoupled design limits joint optimization with the backbone. VoxelDet [20] applies a training-free hierarchical pruning for two-stage detectors, but its heuristic design easily results in inaccurate pruning decisions and unsuitable pruning intensity. In summary, existing methods depend on dense intermediate representations, lack end-to-end integration with sparse backbones, or are task-specific. We implement progressive backbone pruning via a tailored strategy that ensures precise background elimination and efficient reduction.

2.3 Center Prediction and Sparse Peak Selection

Detection heads using object centers as output units can be divided into dense and sparse heads, which differ in candidate generation and duplicate suppression. The dense paradigm originates in CenterNet [3], which extracts object centers from Gaussian heatmaps via max pooling; CenterPoint [2] extends it to 3D BEV detection by projecting sparse voxel features onto dense maps. Transfusion [11] and DSVT [9] retain it as a de facto standard. This mechanism inherently depends on sparse-to-dense conversion, conflicting with the efficiency objective of purely sparse architectures. To remove this overhead, VoxelNeXt [7] and the FSD series [5, 6] propose purely sparse heads. VoxelNeXt employs a voxel-to-object scheme with pseudo-center regression, and FSD v2 [6] adopts center voting with virtual voxels. However, purely sparse architectures lack dense heatmaps' local peak selection and instead rely on global top-k followed by NMS, yielding unstable deduplication. To fill this gap, we propose a Dynamic Transfusion Head that emulates the dense BEV peak mechanism via dynamic pooling within predicted bounding boxes, providing a native and general proposal generation and duplicate-suppression scheme.

3 Methodology

3.1 Overall Architecture

In this section, the proposed 3D object detection method named DeS3 is introduced in detail. Fig. 1 illustrates the overall architecture of DeS3. Specifically, we design a cascaded progressive pruning network composed of multiple lightweight UNets as the backbone of the detector. After each UNet in the pruning network, a Select and Drop module is introduced to progressively discard low-confidence points in a layer-wise manner, gradually transforming the initial sparse voxel set into an extremely compact super sparse space. Following prior works [5, 6], a virtual voxel generation module is appended to mitigate surface-only localization errors. The fused features are then fed into the Dynamic Transfusion Head, which performs bounding box regression and classification on the preserved high-value voxels to yield precise predictions. It is worth noting that all UNets are implemented with a Sparse Transformer (SpTransformer) based on Sparse Matrix Multiplication (SPMM) that enables efficient multi-scale feature extraction and fusion directly on sparse voxels.

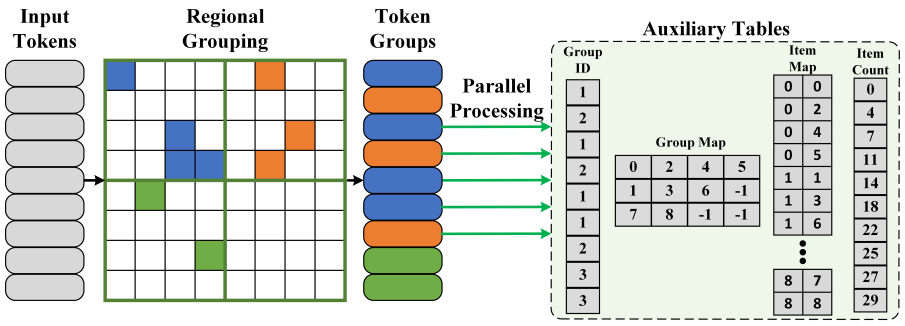

Figure 2: Workflow of regional grouping and auxiliary tables construction for SPMM-based self-attention.

3.2 Sparse Matrix Multiplication and Sparse Transformers

Existing Transformers applied in sparse voxel space take sparse tensors as input, yet their self-attention relies on fixed-length zero-padding to fit dense kernels. This paradigm incurs non-negligible computational redundancy at padded positions within highly sparse windows. To bridge this gap, we transfer the full computation flow into operator-level purely sparse Sparse Matrix Multiplication (SPMM) operations.

3.2.1 Regional Grouping and Auxiliary Table Construction

The foundation of our SPMM-based self-attention lies in the regional partitioning of input voxel tokens. Inspired by the Swin Transformer [21], we decompose global attention into non-overlapping local windows, assign voxels in each window to the same group, and conduct self-attention strictly within each local group (Fig. 2). Following regional grouping, we construct four auxiliary tables that jointly define the sparse connectivity pattern within each group and support subsequent sparse matrix multiplication.

Group ID: A 1D tensor mapping each voxel to its corresponding group index, serving as the primary index for group assignment.

Group Map: A 2D tensor where each row records the positional indices of all voxels belonging to a specific group, enabling fast localization of intra-group members.

Item Map: A flattened array of (Query, Key) index pairs for all attention entries, defining the fully-connected intra-group sparse pattern and the positions of non-zero elements.

Item Count: The cumulative sum of per-voxel attention entries, locating the start and end positions of each voxel's attention segment within the Item Map.

For cross-window interaction, we adopt the shifted window mechanism [21] in alternating layers. Both standard and shifted window partitions share the same auxiliary table pipeline.

3.2.2 SPMM-based Self-Attention

Based on the sparse connectivity pattern defined by the auxiliary tables, the computation of SPMM-based self-attention is completed in two stages, as shown in Fig. 3. In the first stage, a sparse matrix multiplication is performed on $\mathbf{Q}$ and $\mathbf{K}^\top$ strictly following the (Query, Key) pairs in the Item Map to obtain the sparse attention weight matrix $\mathbf{Attentions}$. In the second stage, another sparse matrix multiplication is performed between the attention weight matrix $\mathbf{Attentions}$ and the value matrix $\mathbf{V}$ to obtain the attention output. To facilitate this computation, the **Item Count** is utilized to dynamically locate the start and end indices of valid attention entries for each Query. Guided by these indices, the operator retrieves the corresponding normalized attention weights from $\mathbf{Attentions}$. Meanwhile, the associated Value features are obtained from $\mathbf{V}$ via the **Group ID** and **Group Map** to perform a weighted summation. Eventually, the aggregated features are scattered back to their original spatial positions in the output tensor.

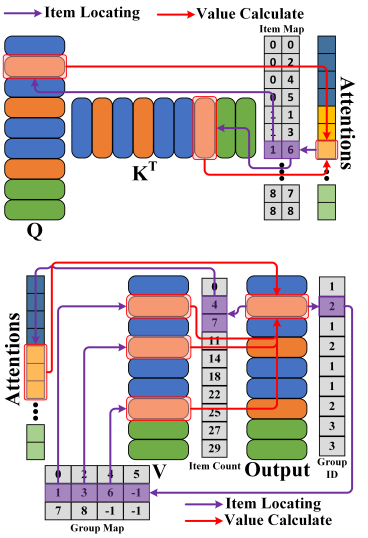

Figure 3: Two-stage computation workflow of SPMM sparse attention.

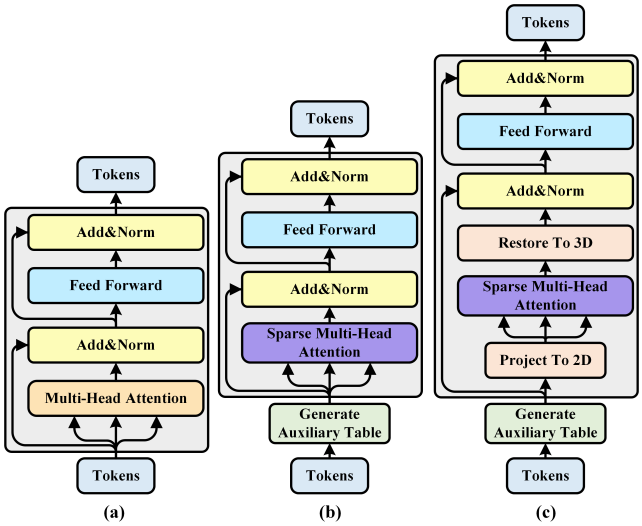

Figure 4: Structural comparison of three Transformer blocks. (a) Standard Transformer block, (b) Pure-mode SpTransformer block, (c) Mix-mode SpTransformer block.

3.2.3 SPARSE TRANSFORMER BLOCK

Building on the SPMM-based self-attention, we construct a Sparse Transformer (SpTransformer) block. To balance capacity and overhead across network depths, we design two configuration modes, namely the Pure and Mix modes (Fig. 4). The Pure mode performs self-attention and feed-forward in a unified dimensional space, branching into Pure 3D and Pure 2D. The Mix mode restricts self-attention on the 2D pillar plane to constrain quadratic complexity growth, while keeping the feed-forward in the 3D voxel space to retain spatial details, yielding an optimal accuracy-efficiency balance. In DeS3, all UNets are constructed by SpTransformer in Mix mode.

3.3 PROGRESSIVE PRUNING NETWORK

Beyond efficient attention computation, reducing the number of voxels that participate in computation is equally important. Existing detectors process all non-empty voxels indiscriminately, incurring severe redundancy since background voxels dominate valid voxels. We embed a progressive pruning paradigm into the sparse backbone for end-to-end joint training. Each stage of the backbone has a lightweight voxel importance predictor that estimates per-voxel foreground probability, and low-confidence background voxels are pruned layer by layer via a threshold strategy, yielding stepwise reduction in computational load as depth increases. The remaining voxels form an extremely sparse, foreground-dominated super sparse space.

The core challenge of the progressive pruning paradigm lies in balancing pruning efficiency against detection accuracy. Retaining too many background voxels limits effectiveness, while falsely pruning foreground voxels causes missed detections. To guarantee safe and effective pruning, we define three fundamental principles and construct a dedicated loss function to implement them via training supervision. The three principles are as follows.

Principle 1: Early-Stage Pruning. Discard background voxels as early as possible, since voxels pruned at shallow stages are excluded from all subsequent layers and yield the greatest cumulative savings.

Principle 2: Strict Foreground Preservation. Falsely pruning foreground voxels causes missed detections, the most severe failure mode. We therefore treat foreground retention as the primary constraint, even at the cost of retaining some ambiguous background voxels.

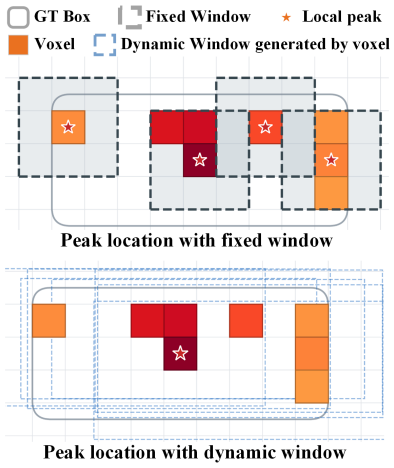

Figure 5: Comparison of fixed-window and dynamic-window peak localization in sparse voxel scenarios.

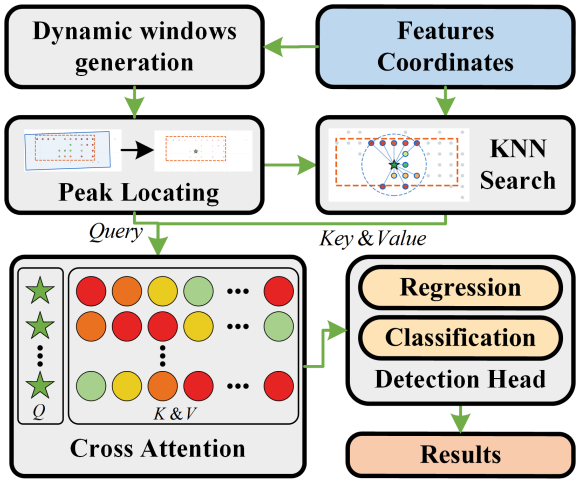

Figure 6: Overall architecture of the Dynamic Transfusion Head.

Principle 3: Complexity Adaptation. A background voxel surviving pruning at both stages k and $k+1$ is a hard sample for the stage-k classifier; excessive loss pressure on such samples may cause overfitting and instability. Therefore, the loss contribution of such samples at shallower stages should be adaptively down-weighted based on their deeper-stage predictions.

To translate the three principles into quantitatively enforceable training constraints, we construct a modulated multi-stage pruning loss that instantiates each principle via dedicated per-sample weight mechanisms. We provide the full derivation of the pruning loss and the four modulation strategies in Appendix A.

3.4 Dynamic Transfusion Head

Purely sparse architectures inherently lack the peak selection mechanism of dense BEV heatmaps, where local maxima correspond to object centers. Unlike prior sparse works using global top-k followed by NMS [7] or voting [5, 6], we propose a Dynamic Transfusion Head that simulates the dense peak mechanism on purely sparse voxels via dynamic pooling within predicted bounding boxes. The motivation stems from the difference between the two data distributions. Dense heads benefit from spatially continuous BEV features, where fixed-window max pooling suppresses non-peak regions to identify local maxima. On purely sparse voxel sets, however, the discrete distribution can leave a fixed window with no neighboring voxels, making traditional max pooling inapplicable.

To address this, we design a dynamic window peak localization strategy. Specifically, it uses the bounding box predicted by each voxel as a dynamic pooling window, and a voxel is a peak if and only if its classification score is the maximum within its own predicted box. As shown in Fig. 5, this mitigates the multi-peak artifact induced by sparse data by reproducing dense heatmaps' local-maximum principle on sparse voxels.

As shown in Fig. 6, the Dynamic Transfusion Head takes features and coordinates as input and operates in two stages: the first produces queries via dynamic window generation and peak localization; the second retrieves K-nearest neighbors as key-value pairs and refines query features via cross-attention, producing 3D detections.

Table 1: Comparison with the state-of-the-art detectors on the nuScenes validation split. Abbreviations: Trailer (T.L.), Construction Vehicle (C.V.), Pedestrian (Ped.), Motorcycle (M.T.), Traffic Cone (T.C.), and Barrier (B.R.).

Method	Head Type	NDS	mAP	Car	Truck	Bus	T.L.	C.V.	Ped.	M.T.	Bike	T.C.	B.R.
VISTA[23]	Dense	69.5	62.8	85.9	60.7	68.4	41.4	23.5	85.4	70.2	55.5	71.5	65.8
CenterPoint[2]	Dense	66.5	59.2	84.9	57.4	70.7	38.1	16.9	85.1	59.0	42.0	69.8	68.3
Transfusion-L[11]	Dense	70.1	65.5	86.9	60.8	73.1	43.4	25.2	87.5	72.9	57.3	77.2	70.3
PillarNeXt[24]	Dense	68.4	62.2	85.0	57.4	67.6	35.6	20.6	86.8	68.6	53.1	77.3	69.7
HEDNet[25]	Dense	71.4	66.7	87.7	60.6	77.8	50.7	28.9	87.1	74.3	56.8	76.3	66.9
DSVT[9]	Dense	71.1	66.4	87.4	62.6	75.9	42.1	25.3	88.2	74.8	58.7	77.8	70.9
PillarNeSt[26]	Dense	71.6	66.9	87.4	56.4	64.0	63.0	30.7	86.6	69.4	51.5	82.3	77.9
Voxel Mamba[27]	Dense	71.9	67.5	87.9	62.8	76.8	45.9	24.9	89.3	77.1	58.6	80.1	71.5
VoxelNeXt[7]	Sparse	68.7	63.5	83.9	55.5	70.5	38.1	21.1	84.6	62.8	50.0	69.4	69.4
FSDv2[6]	Sparse	70.4	64.7	84.4	57.3	75.9	44.1	28.5	86.9	69.5	57.4	72.9	73.6
SAFDNet[28]	Sparse	71.0	66.3	87.6	60.8	78.0	43.5	26.6	87.8	75.5	58.0	75.0	69.7
DeS3 (Ours)	Sparse	**72.4**	**67.9**	87.3	64.9	77.8	44.4	25.9	88.4	76.7	63.2	75.4	75.1

4 EXPERIMENTS

4.1 DATASETS AND EVALUATION METRICS

We evaluate on the nuScenes benchmark [12], which contains 1,000 scenes with 28,130 training, 6,019 validation, and 6,008 test samples over 10 classes. Following the official protocol [12], we use mean Average Precision (mAP) and nuScenes Detection Score (NDS) as primary metrics. Ablation studies are on the validation set; test-set results are obtained via submission to the official server.

4.2 IMPLEMENTATION DETAILS

Following prior works [2, 5, 8, 9], we use 10-frame accumulated LiDAR sweeps with a voxel size of 0.2 m. The detector training is performed on 4 NVIDIA RTX A6000 GPUs with a batch size of 24 for 10 epochs. Although this training schedule is shorter than the 20 epochs of classic methods, our fully sparse architecture and foreground-focused pruning ensure sufficient convergence. We use Adam with a one-cycle policy [22] and an initial learning rate of 0.0015, while other training and inference settings follow DSVT [9]. Latency and GPU memory are measured on a single A6000 GPU with a batch size of 1. Besides, the hyperparameters of pruning loss are detailed in Appendix A.

4.3 COMPARISON TO STATE-OF-THE-ART METHODS

We begin our experimental evaluation by comparing DeS3 with state-of-the-art LiDAR-only detectors on the nuScenes validation set. Table 1 reports overall metrics (mAP and NDS) and per-category AP over 10 classes. The compared methods are grouped into two categories based on their detection head design. The dense-head methods project sparse features onto dense BEV grids, while sparse-head methods operate entirely on sparse voxels. All methods use the LiDAR-only, 10-sweep setting, with results sourced from original publications, official repositories, or our reproduction under a unified framework.

As shown in Table 1, DeS3 achieves 72.4% NDS and 67.9% mAP on the nuScenes validation set, outperforming all compared methods. At the per-category level, DeS3 achieves the best results on Truck and Bike, outperforming the second-best methods by 2.1 and 4.5 percentage points, respectively. Additionally, DeS3 also delivers competitive second-place results on Pedestrian, Motorcycle, and Bus, while remaining competitive on Car, Barrier, and Traffic Cone. Moreover, we further verify the generalization ability of DeS3 on the nuScenes test split, where our method also attains state-of-the-art performance (73.8% NDS and 69.3% mAP). The detailed comparison and per-category analysis are provided in Appendix B.

Beyond detection accuracy, inference latency and GPU memory consumption are also critical for assessing practical deployment feasibility. Fig. 7 compares our method with FSDv2, VoxelNeXt, SAFDNet, and DSVT on a single A6000 GPU. DeS3 has the lowest peak memory (469.2 MB, 33.7% of DSVT) but slightly higher latency (173.04 ms). Overall, DeS3 maintains a competitive accuracy-memory-speed trade-off, making it suitable for high-accuracy perception tasks in resource-constrained autonomous driving.

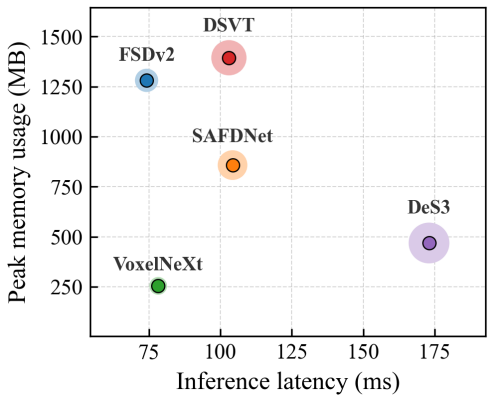

Figure 7: Efficiency comparison of different methods at end-to-end level.

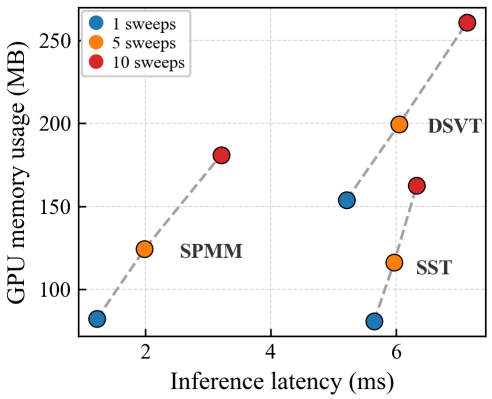

Figure 8: Efficiency comparison of different sparse attention mechanisms at operator level.

Table 2: Performance comparison of backbone variants on the nuScenes validation split.

Type	mAP	NDS	MAC
2D	66.4	71.2	170.9 G
3D	67.6	72.1	454.0 G
Mix	**67.9**	**72.4**	389.5 G

Table 3: Performance comparison of pruning strategies on the nuScenes validation split.

No.	SFP	CA	ESP	mAP	NDS	PSDR	NSDR
P1	✓	✓	✓	**67.9**	**72.4**	0.12%	83.10%
P2		✓	✓	65.3	70.6	1.21%	95.71%
P3	✓		✓	67.4	72.1	0.13%	82.60%
P4	✓	✓		67.7	72.4	0.11%	80.55%

4.4 Ablation Studies

Efficiency of SPMM: To quantitatively verify the efficiency advantage of the proposed SPMM, we benchmark it against two representative sparse window attention implementations, including SST [8] (zero-padded buckets) and DSVT [9] (rotating sets). All experiments use a voxel size of 0.2 m, and a window size of 6×6. The SST employs 5-level capacity bucketing 8,16,32,64,128, the DSVT uses set size 20, and the SPMM is implemented as a custom CUDA kernel. We evaluate on the first 20 frames of the nuScenes validation set with 1/5/10-sweeps inputs. Fig. 8 reports the quantitative comparison of inference time (including grouping and computation) and peak GPU memory usage. In terms of inference time, SPMM consistently achieves the lowest latency across all configurations. As the number of sweeps increases from 1 to 10, SPMM maintains more than 2× speedup over both SST and DSVT. Regarding memory consumption, DSVT consistently exhibits the largest footprint among the three methods, which we attribute to its fixed-size padded tensors. Meanwhile, SPMM and SST exhibit comparable memory usage at low sweep counts, both substantially lower than DSVT. However, SPMM's memory usage grows slightly faster than SST as sweeps increase, yet both remain far below DSVT.

Performance of Backbone Variants: After verifying the efficiency of SPMM, we investigate how the computation dimension affects detection performance and computational cost (via Multiply-Accumulate Operations, MACs). As shown in Table 2, the pure 2D mode minimizes cost but sacrifices accuracy due to limited 3D context, while the pure 3D mode improves accuracy at excessive computational cost. The Mix mode achieves the best accuracy with 14.2% lower cost than the pure 3D mode, striking the optimal balance. This advantage arises from its dimension-decoupled architecture. The 2D Self-Attention achieves efficient horizontal semantic aggregation with limited computational cost, while avoiding the disruption of vertically continuous features introduced by strict 3D window partitioning. The 3D per-voxel FFN further retains fine-grained spatial structural information.

Effect of Different Pruning: Progressive pruning relies on three synergistic principles including Strict Foreground Preservation (SFP), Complexity Adaptation (CA), and Early-Stage Pruning (ESP). In this section, we conduct ablation studies to evaluate each principle's individual contribution and its impact on accuracy and efficiency, as shown in Table 3. In addition to mAP and NDS, we

Table 4: Performance comparison of pruning stages on the nuScenes validation split.

No.	Pruning stages	mAP	NDS	PSDR	NSDR
S1	2	67.5	72.3	0.09%	77.93%
S2	3	**67.9**	**72.4**	0.12%	83.10%
S3	4	67.2	71.9	0.15%	85.76%

Table 5: Performance comparison of detection head variants on the nuScenes validation split.

Method	mAP	NDS
VoxelNeXt Head[7]	67.2	72.2
Transfusion Head[11]	*Not Converged*	
Dynamic Transfusion Head	**67.9**	**72.4**

introduce two voxel-level metrics, Positive Sample Dropout Rate (PSDR) and Negative Sample Dropout Rate (NSDR), to reveal underlying mechanisms. Both PSDR and NSDR are computed based on the voxel dropout and retention statistics after the pruning procedure. Removing SFP (P1 vs P2) causes the largest degradation among all principles. At the voxel level, both PSDR and NSDR increase substantially, reflecting that the network shifts to aggressive, indiscriminate pruning. The loss of foreground structural information directly causes the significant drop in detection accuracy. Dropping CA (P1 vs P3) also causes mAP/NDS degradation with subtle PSDR/NSDR variation. This is because CA adaptively down-weights hard, ambiguous samples across layers, preventing premature aggressive pruning and stabilizing pruning decisions. Removing ESP (P1 vs P4) causes negligible accuracy change, yet the stage-wise distribution of NSDR reveals its contribution (78.6%, 2.6%, 1.9% vs 68.4%, 6.5%, 5.7%). This distribution gap confirms that ESP concentrates the majority of background removal at shallow stages, freeing deep layers to focus on high-confidence foreground regions.

Beyond validating the progressive pruning strategies, we further investigate the effect of pruning stages. As shown in Table 4, S1 removes too few background voxels (lowest NSDR), leaving noise for deep layers and limiting accuracy. S3 delivers the strongest background filtering, but its noticeably higher PSDR inevitably degrades detection accuracy. S2 strikes the optimal trade-off, ensuring both foreground safety and sufficient background removal.

Performance of Detection Head Variants: To justify the effectiveness of the Dynamic Transfusion Head, we conduct a systematic ablation to compare the performance differences brought by three detection heads, as shown in Table 5. VoxelNeXt Head [7] is a single-stage sparse detection head that selects candidates via global top-k. Transfusion Head [11] is a two-stage detection head that first extracts peaks via fixed-window max-pooling and then refines them through cross-attention.

The two baseline detection heads suffer from distinct inherent drawbacks. Directly transplanting the Transfusion Head into sparse voxel space leads to complete training divergence, as fixed-window max-pooling generates massive duplicate peaks for easy objects that suppress proposals of hard samples and break positive sample matching. Meanwhile, VoxelNeXt Head relies on global top-k selection without local spatial suppression, retaining redundant voxels for the same instance. Our dynamic-window peak localization addresses both issues and presents the best performance.

5 Conclusion

In this study, we present a fully sparse 3D object detector, named DeS3, to systematically tackle the efficiency and accuracy bottlenecks for autonomous driving perception. Specifically, an SPMM-based Sparse Transformer is designed to perform zero-redundancy window attention directly on non-empty voxels, eliminating padding-induced computational and memory overhead at the operator level. A progressive pruning network embedded in the backbone hierarchically filters background voxels, achieving stepwise computation reduction while strictly preserving foreground recall. Furthermore, a Dynamic Transfusion Head is introduced to replicate the peak-selection mechanism of dense heatmaps via bounding-box-aware dynamic pooling, mitigating duplicate candidate issues in sparse detection paradigms. Extensive experiments on the nuScenes benchmark demonstrate that DeS3 achieves state-of-the-art performance among sparse-head and dense-head detectors. The results validate that fully sparse architectures can maintain high detection accuracy while scaling computation linearly with valid point volume rather than global perception range, showing great potential for long-range on-board perception systems. Although DeS3 achieves significant progress in accuracy and memory efficiency, the complexity of the proposed architecture inevitably incurs a certain reduction in inference speed. Therefore, future work will prioritize simplifying the network structure and optimizing the end-to-end inference pipeline to achieve more efficient detection.

Reproducibility Statement

To ensure reproducibility, we will release the source code and trained models in a repository upon acceptance. The experimental setup, hyperparameters, and evaluation protocol are described in the implementation details subsection and the appendix, and the nuScenes dataset is publicly available.

Ethics Statement

This work raises no ethics concerns under the ICLR Code of Ethics: it uses only publicly available LiDAR driving datasets and involves no human subjects, no personally identifiable information, and no potentially harmful insights.

AI Use Statement

In this work, we did not employ generative AI tools for research ideation, experimental design, or core algorithmic development. We acknowledge, however, that AI tools were used for literature retrieval and discovery, as well as for assisting in the scripting of figure generation(the efficiency comparisons presented in Figs. 7 and 8). We assume full responsibility for the final content of this paper, including all text, figures, and scientific claims.

References

[1] Y. Yan, Y. Mao, and B. Li, "Second: Sparsely embedded convolutional detection," *Sensors*, vol. 18, no. 10, pp. 3337, 2018.

[2] T. Yin, X. Zhou, and P. Krähenbühl, "Center-based 3d object detection and tracking," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, pp. 11784–11793, 2021.

[3] X. Zhou, D. Wang, and P. Krähenbühl, "Objects as points," *arXiv preprint arXiv:1904.07850*, 2019.

[4] J. Mao, Y. Xue, M. Niu, H. Bai, J. Feng, X. Liang, H. Xu, and C. Xu, "Voxel transformer for 3d object detection," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, pp. 3144–3153, 2021.

[5] L. Fan, F. Wang, N. Wang, and Z.-X. Zhang, "Fully sparse 3d object detection," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 35, pp. 351–363, 2022.

[6] L. Fan, F. Wang, N. Wang, and Z. Zhang, "Fsd v2: Improving fully sparse 3d object detection with virtual voxels," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 47, no. 2, pp. 1279–1292, 2024.

[7] Y. Chen, J. Liu, X. Zhang, X. Qi, and J. Jia, "Voxelnext: Fully sparse voxelnet for 3d object detection and tracking," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, pp. 21674–21683, 2023.

[8] L. Fan, Z. Pang, T. Zhang, Y.-X. Wang, H. Zhao, F. Wang, N. Wang, and Z. Zhang, "Embracing single stride 3d object detector with sparse transformer," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, pp. 8448–8458, 2022.

[9] H. Wang, C. Shi, S. Shi, M. Lei, S. Wang, D. He, B. Schiele, and L. Wang, "Dsvt: Dynamic sparse voxel transformer with rotated sets," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, pp. 13520–13529, 2023.

[10] Y.-Q. Yang, Y.-X. Guo, J.-Y. Xiong, Y. Liu, H. Pan, P.-S. Wang, X. Tong, and B. Guo, "Swin3d: A pretrained transformer backbone for 3d indoor scene understanding," *Computational Visual Media*, vol. 11, no. 1, pp. 83–101, 2025.

[11] X. Bai, Z. Hu, X. Zhu, Q. Huang, Y. Chen, H. Fu, and C.-L. Tai, "Transfusion: Robust lidar-camera fusion for 3d object detection with transformers," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, pp. 1090–1099, 2022.

[12] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom, "nuscenes: A multimodal dataset for autonomous driving," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, pp. 11618–11628, 2020.

[13] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, pp. 652–660, 2017.

[14] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 30, 2017.

[15] Y. Zhou and O. Tuzel, "Voxelnet: End-to-end learning for point cloud based 3d object detection," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, pp. 4490–4499, 2018.

[16] S. Shi, L. Jiang, J. Deng, Z. Wang, C. Guo, J. Shi, X. Wang, and H. Li, "PV-RCNN++: Point-voxel feature set abstraction with local vector representation for 3D object detection," *Int. J. Comput. Vis.*, vol. 131, no. 2, pp. 531–551, 2023.

[17] S. Shi, Z. Wang, J. Shi, X. Wang, and H. Li, "From points to parts: 3d object detection from point cloud with part-aware and part-aggregation network," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 43, no. 8, pp. 2647–2664, 2020.

[18] T. Zhao, X. Ning, K. Hong, Z. Qiu, P. Lu, Y. Zhao, L. Zhang, L. Zhou, G. Dai, and H. Yang, "Ada3d: Exploiting the spatial redundancy with adaptive inference for efficient 3d object detection," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, pp. 17682–17692, 2023.

[19] J. Li, T. Sun, X. Zhang, Z. Wang, B. Feng, and K. Xu, "Efficient 3D perception on multi-sweep point cloud with Gumbel spatial pruning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, pp. 15726–15732, 2025.

[20] J. Wen, J. Li, and T. Zhang, "VoxelDet: Towards accurate 3D object detection with voxel pruning and fine geometric shape," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, pp. 1–6, 2025.

[21] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo, "Swin transformer: Hierarchical vision transformer using shifted windows," in *Proc. IEEE/CVF Int. Conf. Comput. Vis. (ICCV)*, pp. 9992–10002, 2021.

[22] L. N. Smith and N. Topin, "Super-convergence: Very fast training of neural networks using large learning rates," in *Proc. Int. Conf. Artif. Intell. Appl. Innov.*, vol. 11006, pp. 369–386, 2019.

[23] S. Deng, Z. Liang, L. Sun, and K. Jia, "Vista: Boosting 3d object detection via dual cross-view spatial attention," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, pp. 8438–8447, 2022.

[24] J. Li, C. Luo, and X. Yang, "PillarNeXt: Rethinking network designs for 3D object detection in LiDAR point clouds," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, pp. 17567–17576, 2023.

[25] G. Zhang, C. Junnan, G. Gao, J. Li, and X. Hu, "Hednet: A hierarchical encoder-decoder network for 3d object detection in point clouds," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 36, pp. 53076–53089, 2023.

[26] W. Mao, T. Wang, D. Zhang, J. Yan, and O. Yoshie, "Pillarnest: Embracing backbone scaling and pretraining for pillar-based 3d object detection," *IEEE Trans. Intell. Veh.*, vol. 10, no. 4, pp. 2905–2914, 2024.

[27] G. Zhang, L. Fan, C. He, Z. Lei, Z. Zhang, and L. Zhang, "Voxel mamba: Group-free state space models for point cloud based 3d object detection," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 37, pp. 81489–81509, 2024.

[28] G. Zhang, J. Chen, G. Gao, J. Li, S. Liu, and X. Hu, "Safdnet: A simple and effective network for fully sparse 3d object detection," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, pp. 14477–14486, 2024.

[29] A. H. Lang, S. Vora, H. Caesar, L. Zhou, J. Yang, and O. Beijbom, "Pointpillars: Fast encoders for object detection from point clouds," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, pp. 12689–12697, 2019.

[30] Z. Yang, Y. Sun, S. Liu, and J. Jia, "3dssd: Point-based 3d single stage object detector," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, pp. 11037–11045, 2020.

[31] Z. Tian, X. Chu, X. Wang, X. Wei, and C. Shen, "Fully convolutional one-stage 3d object detection on lidar range images," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 35, pp. 34899–34911, 2022.

[32] Y. Hu, Z. Ding, R. Ge, W. Shao, L. Huang, K. Li, and Q. Liu, "Afdetv2: Rethinking the necessity of the second stage for object detection from point clouds," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, vol. 36, pp. 969–979, 2022.

[33] Y. Chen, Y. Li, X. Zhang, J. Sun, and J. Jia, "Focal sparse convolutional networks for 3d object detection," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, pp. 5418–5427, 2022.

[34] Y. Li, Y. Chen, X. Qi, Z. Li, J. Sun, and J. Jia, "Unifying voxel-based representation with transformer for 3d object detection," in *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 35, pp. 18442–18455, 2022.

[35] Y. Chen, J. Liu, X. Zhang, X. Qi, and J. Jia, "Largekernel3d: Scaling up kernels in 3d sparse cnns," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, pp. 13488–13498, 2023.

[36] G. Shi, R. Li, and C. Ma, "Pillarnet: Real-time and high-performance pillar-based 3d object detection," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, pp. 35–52, 2022.

[37] T. Lu, X. Ding, H. Liu, G. Wu, and L. Wang, "Link: Linear kernel for lidar-based 3d perception," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, pp. 1105–1115, 2023.

A Derivation of the Progressive Pruning Loss

The three principles described in the main text are translated into a modulated multi-stage pruning loss that instantiates each principle via dedicated weight mechanisms. For each stage k, the per-voxel pruning loss is:

$$\mathcal{L}_{\text{prune}}^{(k)} = \sum_{i=1}^{N_k} w_i^{(k)} \cdot \text{FL}\left(p_i^{(k)}, y_i^{(k)}\right) \tag{1}$$

where N_k is the voxel count in stage k, $p_i^{(k)}$ the foreground probability of voxel i, $y_i^{(k)} \in \{0, 1\}$ its binary label, and $\text{FL}(\cdot)$ the Focal Loss. The per-sample weight $w_i^{(k)}$ uses distinct modulation strategies for four situations.

Correct predictions: $w_i^{(k)} = 1$. Correctly classified voxels keep standard gradient flow for stable convergence.

Foreground mistakenly pruned: $w_i^{(k)} = \beta$, a large value that enforces **Principle 2 (strict foreground preservation)** by suppressing pruning on foreground voxels.

Background incorrectly retained but pruned next stage: A background voxel misclassified as foreground at stage k but pruned at stage $k+1$ is eventually corrected, yet its retention at k violates early removal. To enforce **Principle 1 (early-stage pruning priority)**, we assign a stage-dependent penalty $w_i^{(k)} = \gamma_k$ that decays from shallow to deep, so retaining background earlier incurs a larger penalty and removal happens as early as possible.

Background retained across multiple stages: If a background voxel survives both stages k and $k+1$, it is a hard sample for the stage-k classifier. Rather than applying the fixed γ_k, we adaptively down-weight its loss at stage k based on its deeper-stage prediction:

$$w_i^{(k)} = \left(1 - p_i^{(k+1)}\right)^{\delta_k} \tag{2}$$

with modulation exponent δ_k. This reduces pressure on ambiguous background patterns, stabilizing training and realizing **Principle 3 (complexity adaptation)**.

The total pruning loss over all stages is:

$$\mathcal{L}_{\text{prune}} = \sum_{k=1}^{K} \alpha_k \cdot \mathcal{L}_{\text{prune}}^{(k)} \tag{3}$$

where K is the number of pruning stages and α_k the stage-wise weight coefficient. α_k decreases from shallow to deep, further enforcing **Principle 1 (early-stage pruning priority)** by strengthening supervision on earlier layers.

Hyperparameters. For the progressive pruning module, all pruning thresholds R are set to 0.2, the foreground penalty β to 128, and the stage-wise weight coefficients $\alpha_{1\sim3}$, stage-dependent background retention penalties $\gamma_{1\sim3}$, and hard sample modulation exponents $\delta_{1\sim3}$ are set to $\{4, 2, 1\}$, $\{4, 2, 1\}$, and $\{2, 1, 0\}$, respectively.

Ablation configurations. In the ablation studies of the main text, each principle is disabled by adjusting its corresponding weight: disabling SFP sets the foreground penalty β to 1; disabling CA sets the hard-sample modulation exponents $\delta_{1\sim3}$ to 0; disabling ESP sets the stage-wise weight coefficients $\alpha_{1\sim3}$ and stage-dependent background retention penalties $\gamma_{1\sim3}$ to 1.

B Additional Results on the nuScenes Test Split

Table 6: Comparison with the state-of-the-art detectors on the nuScenes test split. Abbreviations: Trailer (T.L.), Construction Vehicle (C.V.), Pedestrian (Ped.), Motorcycle (M.T.), Traffic Cone (T.C.), and Barrier (B.R.).

Method	Head Type	NDS	mAP	Car	Truck	Bus	T.L.	C.V.	Ped.	M.T.	Bike	T.C.	B.R.
PointPillars[29]	Dense	45.3	30.5	68.4	23.0	28.2	23.4	4.1	59.7	27.4	1.1	30.8	38.9
3DSSD[30]	Dense	56.4	42.6	81.2	47.2	61.4	30.5	12.6	70.2	36.0	8.6	31.1	47.9
FCOS-L[31]	Dense	65.7	60.2	82.2	47.7	52.9	48.8	28.8	84.5	68.0	39.0	79.2	70.7
CenterPoint[2]	Dense	65.5	58.0	84.6	51.0	60.2	53.2	17.5	83.4	53.7	28.7	76.7	70.9
AFDetV2[32]	Dense	68.5	62.4	86.3	54.2	62.5	58.9	26.7	85.8	63.8	34.3	80.1	71.0
VISTA[23]	Dense	69.8	63.0	84.4	55.1	63.7	54.2	25.1	82.8	70.0	45.4	78.5	71.4
Focals Conv[33]	Dense	70.0	63.8	86.7	56.3	67.7	59.5	23.8	87.5	64.5	36.3	81.4	74.1
UVTR-L[34]	Dense	69.7	63.9	86.3	52.2	62.8	59.7	33.7	84.5	68.8	41.1	74.7	74.9
LargeKernel3D[35]	Dense	70.6	65.4	85.5	53.8	64.4	59.5	29.7	85.9	72.7	46.8	79.9	75.5
Transfusion-L[11]	Dense	70.2	65.5	86.2	56.7	66.3	58.8	28.2	86.1	68.3	44.2	82.0	78.2
PillarNet[36]	Dense	71.4	66.0	87.6	57.5	63.6	63.1	27.9	87.3	70.1	42.3	83.3	77.2
LinK[37]	Dense	71.0	66.3	86.1	55.7	65.7	62.1	30.9	85.8	73.5	47.5	80.4	75.5
HEDNet[25]	Dense	72.0	67.7	87.1	56.5	70.4	63.5	33.6	87.9	70.4	44.8	85.1	78.1
DSVT[9]	Dense	72.7	68.4	86.8	58.4	67.3	63.1	37.1	88.0	73.0	47.2	84.9	78.4
Voxel Mamba[27]	Dense	73.0	69.0	86.8	57.1	68.0	63.2	35.4	89.5	74.7	50.8	86.9	77.3
FSDv2[6]	Sparse	71.7	66.2	83.7	51.6	66.4	59.1	32.5	87.1	71.4	51.7	80.3	78.7
VoxelNeXt[7]	Sparse	70.0	64.5	84.6	53.0	64.7	55.8	28.7	85.8	73.2	45.7	79.0	74.6
SAFDNet[28]	Sparse	72.3	68.3	87.3	57.3	68.0	63.7	37.3	89.0	71.1	44.8	84.9	79.5
DeS3 (Ours)	Sparse	**73.8**	**69.3**	86.7	58.0	67.5	64.3	32.2	88.4	77.9	55.4	82.6	79.8

To further verify the generalization ability of DeS3, we report test-set performance on the nuScenes benchmark in Table 6. Test set results are obtained via submission to the official evaluation server, and all baseline results are sourced from the original publications or the public leaderboard. Specifically, the proposed DeS3 achieves 73.8% NDS and 69.3% mAP on the nuScenes test split, establishing a new state of the art among all compared LiDAR-only detectors. Compared with the top sparse-head baseline SAFDNet, DeS3 delivers 1.5% NDS and 1.0% mAP improvements, breaking the inherent accuracy-efficiency trade-off in fully sparse paradigms. Moreover, our method also outperforms leading dense-head detectors including Voxel Mamba and DSVT, proving that fully sparse architectures can achieve superior accuracy while retaining linear computational complexity,

rather than sacrificing precision for efficiency. Meanwhile, per-category results show DeS3 delivers consistently competitive performance across most classes, with particularly notable advantages on small, slender agents such as motorcycles and bikes. Performance on rare classes such as construction vehicles still has room for improvement, consistent with validation set trends. Overall, these test-set results are consistent with the validation results, strongly confirming the feasibility in improving detection performance within the sparse detection paradigm.