# Toward Optimal Mixture of Experts System for 3D Object Detection: A Game of Accuracy, Efficiency and Adaptivity

Linshen Liu, Pu Wang, Guanlin Wu, Junyue Jiang, and Hao Frank Yang

***Abstract*—Autonomous vehicles, open-world robots, and other automated systems rely on accurate, efficient perception modules for real-time object detection. Although high-precision models improve reliability, their processing time and computational overhead can hinder real-time performance and raise safety concerns. This paper introduces an *Edge-based Mixture-of-Experts Optimal Sensing* (*EMOS*) System that addresses the challenge of co-achieving accuracy, latency and scene adaptivity, further demonstrated in the open-world autonomous driving scenarios. Algorithmically, *EMOS* fuses multimodal sensor streams via an Adaptive Multimodal Data Bridge and uses a scenario-aware MoE switch to activate only a complementary set of specialized experts as needed. The proposed hierarchical backpropagation and a multiscale pooling layer let model capacity scale with real-world demand complexity. System-wise, an edge-optimized runtime with accelerator-aware scheduling (e.g., ONNX/TensorRT), zero-copy buffering, and overlapped I/O–compute enforces explicit latency/accuracy budgets across diverse driving conditions. Experimental results establish *EMOS* as the new state of the art: on KITTI, it increases average AP by 3.17% while running 2.6× faster on Nvidia Jetson. On nuScenes, it improves accuracy by 0.2% mAP and 0.5% NDS, with 34% fewer parameters and a 15.35× Nvidia Jetson speedup. Leveraging multimodal data and intelligent experts cooperation, *EMOS* delivers accurate, efficient and edge-adaptive perception system for autonomous vehicles, thereby ensuring robust, timely responses in real-world scenarios.**

***Index Terms*—Mixture of expert (MoE), computing system, efficiency, 3D object detection, edge computing.**

## I. Introduction

DESIGNING machine-learning systems entails an inherent Pareto trade-off: predictive accuracy and computational efficiency cannot, in general, be simultaneously achieved. Nevertheless, real-world deployment demands both in tandem: accuracy high enough to be trustworthy and efficiency tight enough to meet latency and power budgets; consequently, the design task becomes co-optimizing along the Pareto frontier. In autonomous driving systems (ADS), safety hinges on two coupled factors, the fidelity of environment perception and the end-to-end perception-to-decision latency required to meet real-time deadlines. Heterogeneous sensors (LiDAR, cameras, and radar) must be fused under tight compute and memory budgets at the edge; a pipeline that is accurate but slow can miss fast-evolving hazards, whereas one that is fast but inaccurate can precipitate unsafe actuation [1], [2]. The trade-off between accuracy and efficiency is further shaped by sensor bandwidth, field of view, weather robustness, and deployment constraints (e.g., embedded platforms), which together determine what can be processed within a fixed latency budget. This coupled design problem is ubiquitous, spanning robotics, healthcare, and transportation, and thus serves as a general principle for next-generation ML system design [3], [4].

Current computing methods for 3D object detection largely follows two design paradigms. The first develops high-capacity models for accuracy and subsequently retrofits them for deployment via pruning, quantization, operator/kernel fusion, accelerator-aware scheduling, and other systems optimizations to meet latency budgets. The second starts from an efficiency-first stack such as lightweight backbones, streaming/online inference, and low-precision arithmetic (e.g., INT8/FP8) and then attempts to recover accuracy through fine-tuning, self-supervised/distillation objectives, and task-specific adaptation [5], [6], [7]. In practice, neither path yields a stable balance: accuracy-first models often remain too slow as parameter depth/width grows, whereas efficiency-first models typically stagnate below the accuracy required for safety-critical use, shown as Fig. 1. The tension is both computational and architectural. Real-time constraints, memory bandwidth, power limits, and cross-sensor synchronization/fusion limit the speed of per-frame multimodal processing [8]. Architecturally, static pipelines allocate a fixed compute budget regardless of scene difficulty: overcomputing on easy frames and under-provisioning on hard ones [9].

Beyond these challenges, the dynamic operating conditions in ADS make the problem even harder [2], [10]. Open world robotics and autonomous vehicles navigate interactive, rapidly evolving scenes rather than the quasi-stationary settings common in other embodied systems. Dense urban traffic demands fine-grained recognition under frequent occlusions at moderate speeds, whereas suburban highways require long-range

Received 3 March 2025; revised 3 September 2025; accepted 9 September 2025. Date of publication 22 September 2025; date of current version 3 December 2025. Recommended for acceptance by M. Cheng. *(Corresponding author: Hao Frank Yang.)*

Linshen Liu, Pu Wang, Guanlin Wu, and Junyue Jiang are with the Department of Civil and Systems Engineering (System Track), Johns Hopkins University, Baltimore, MD 21218 USA (e-mail: lliu148@jhu.edu; pwang80@jhu.edu; gwu32@jhu.edu; jjiang67@jhu.edu).

Hao Frank Yang is with the Department of Civil and Systems Engineering, Data Science and AI Institute, Johns Hopkins University, Baltimore, MD 21218 USA (e-mail: haofrankyang@jhu.edu).

Data and codes are available at https://github.com/LinshenLiu622/EMOS.

Digital Object Identifier 10.1109/TPAMI.2025.3611795

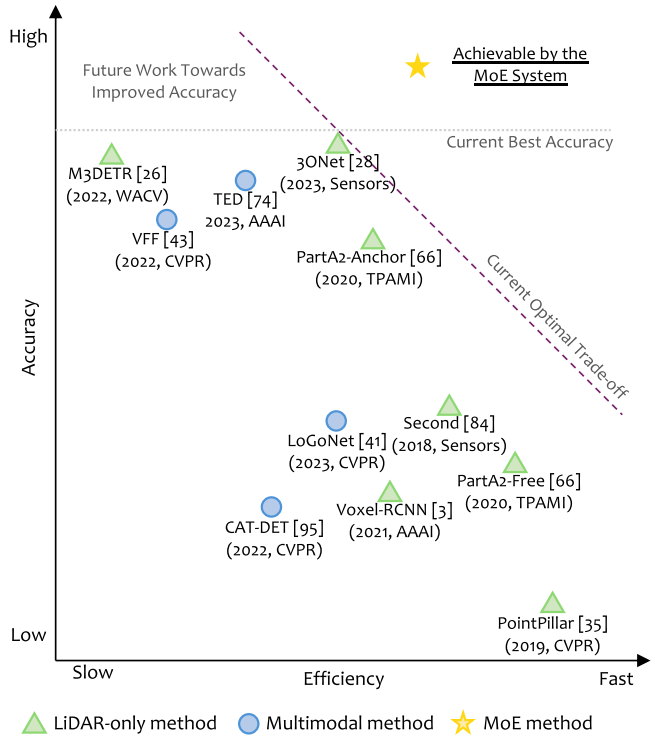



Fig. 1. Performance of 3D object detection models in terms of accuracy and inference efficiency on KITTI and nuScenes. The results indicate that most methods improve accuracy at the expense of inference efficiency. By employing MoE framework, it is possible to break the dilemma between accuracy and efficiency.

perception and timely responses at high speeds. Even along a single route, requirements shift over time: distant agents call for coarse but persistent awareness, while near-field agents require precise, low-latency updates as they approach decision boundaries. Consequently, the optimal balance among accuracy, efficiency, and modality utilization is time-varying, and a single static, unimodal or uniformly multimodal method is unlikely to satisfy all conditions simultaneously [11]. We suggest meeting these demands requires progress on two complementary fronts: (i) leveraging high-fidelity perception method to robustly capture heterogeneous sensor evidence, and (ii) adopting mixture-of-experts (MoE) or related conditional-execution mechanisms to adapt computation to scenario difficulty. Addressing both detection precision and adaptive efficiency enables reliable operation under stringent, safety-critical constraints.

Concretely, the perception front is best addressed by multimodal fusion, which exploits complementary sensing to raise accuracy and robustness in open-world driving scenes [12]. LiDAR offers lighting-invariant 3D geometry but sparse semantics, while cameras provide dense appearance cues at low cost yet degrade under adverse illumination. Fused properly, these modalities compensate for one another: LiDAR anchors spatial precision and camera provides rich semantics, which allows the system to emphasize salient evidence while filtering redundancy and thereby strengthening 3D situational awareness. On the computation front, MoE framework enables conditional execution: instead of a static budget, only a small subset of specialized experts is activated per input, allocating capacity to hard frames and saving cycles on easy ones [13], [14]. MoE's modularity further supports scalability (new experts can be added without retraining the entire model), making it a natural fit for safety-critical perception where accuracy–efficiency requirements vary over time. Deploying MoE for 3D object detection in ADS, however, imposes domain-specific constraints: heterogeneous multi-sensor streaming and data fusion, and tight real-time limitation for inference, which must be explicitly addressed [15].

We propose an ***E**dge-based **M**ixture-of-Experts **O**ptimal **S**ensing* (*EMOS*) System: a system-level solution that not only enhances accuracy through multimodal information but also achieve low inference latency. Different from existing MoE frameworks proposed in Large Language Models (LLMs) [16], [17], [18] and multimodal LLMs (MLLMs) [19], [20], [21], *EMOS* is an adaptive MoE framework for edge-deployable 3D object detection with cooperative experts design. The detailed comparisons are illustrated in Table I. Our contributions can be summarized as follows:

- *A Multimodal MoE Framework with Dynamic Expert Switching.* This work introduces *EMOS*, the first MoE-based 3D object detection framework that jointly addresses the challenges of **accuracy and efficiency** for LiDAR–camera fusion. *EMOS* integrates multimodal sensing with a cooperative mechanism, in which experts of different capacities are incorporated. Large- and small-capacity experts dynamically collaborate, and the MoE router adaptively switches between them based on scenario characteristics. This design enables accurate detection in challenging scenes while keeping inference efficient in simpler environments.
- *Enhanced Accuracy and Robustness via MoE and Hierarchical Training. EMOS* integrates diverse experts: *Efficiency Prioritized Experts* (*EPEs*), *Lightweight Accuracy Prioritized Experts* (*LAPEs*), and *Full-feature Accuracy Prioritized Experts* (*FAPEs*) with a scenario-driven expert-switching mechanism. To mitigate MoE's sensitivity to imbalanced training data, a hierarchical training strategy is proposed, combining intra- and inter-expert backpropagation for collaborative learning. A customized multi-scale pooling layer further enhances efficiency in multimodal processing, ensuring fast and accurate LiDAR-camera fusion.
- *Optimized Edge Inference through System-Level Enhancements.* The original PyTorch-based MoE inference faced inefficiencies in memory management and computational resource allocation, leading to high latency. *EMOS* addresses this through a two-pronged optimization framework, incorporating memory management techniques to increase the L1/L2 cache hit rates and computational graph fusion for reducing redundant data loading across layers, and improving inference speed.
- *State-of-the-art (SOTA) Accuracy and Efficiency Across Benchmarks.* Extensive experiments demonstrate that *EMOS* achieves state-of-the-art performance across both KITTI [10] and nuScenes [2] datasets. On the KITTI, it improves average detection accuracy by 3.17% and runs

TABLE I
COMPARISON BETWEEN EMOS AND EXISTING MOE FRAMEWORKS USED IN LLMS/MLLMS. EMOS IS THE FIRST MOE FRAMEWORK FOR 3D OBJECT DETECTION, INTRODUCING NOVEL DESIGNS IN INPUT MODALITIES, EXPERT SPECIALIZATION, ROUTING, TRAINING STRATEGY, AND SYSTEM OPTIMIZATION BEYOND EXISTING LLM/MLLMS MOE FRAMEWORKS.

| Aspect | EMOS | LLMs/MLLMs MoE Frameworks |
|---|---|---|
| Input Modalities | LiDAR + Camera; Geometric alignment & Adaptive fusion | Text-only (LLMs); Text + Image (MLLMs) |
| Experts | Scenario-aware experts: *EPEs* (High efficiency, LiDAR-only); *APEs* (High accuracy, Multimodal fusion); *Emergency Expert* (Risky cases) | Parameter sub-networks specialized on token-level features without relation to context or scenarios. |
| Routing Mechanism | Adaptive Multimodal Data Bridge & Scenario-aware MoE Switch: 1) Extract features from input LiDAR and image data; 2) Select experts based on distance of proposal regions and confidence; 3) Balance accuracy and efficiency. | Gating network assigns tokens to experts; Balance computational load and ability for scaling. |
| Training Strategy | Hierarchical backpropagation for LiDAR branch and image branch; Long-tail effect mitigation | Auxiliary losses for load balance and avoiding expert collapse; no multimodal hierarchical supervision |
| System Optimization | Optimized for real-world edge deployment: 1) Sparse 3D convolution; 2) Multiscale pooling; 3) Cache-aware memory scheduling | Optimized for cloud-scale GPUs/TPUs: 1) Parallelism (expert/tensor); 2) Distributed training efficiency |

TABLE II
COMPARISON OF EXISTING 3D OBJECT DETECTION METHODS AND EMOS ACROSS ALGORITHM AND SYSTEM DIMENSIONS. UNLIKE PRIOR METHODS, EMOS INTEGRATES MULTIMODAL INPUT, PARAMETER MINIMIZATION, POOLING ENCODER, AND SYSTEM-LEVEL OPTIMIZATIONS, INCLUDING CPU SCHEDULING, MEMORY MANAGEMENT, AND COMPUTATIONAL GRAPHICS OPTIMIZATION, DEMONSTRATING ITS UNIQUE JOINT DESIGN OF ALGORITHM AND SYSTEM.

| Method | Algorithm | | | System | | |
|---|---|---|---|---|---|---|
| | Multimodal Input | Parameter Minimization | Pooling Encoder | CPU Scheduling | Memory Management | Computational Graphic Optimization |
| VoxelNet [22] | ✗ | ✗ | ✓ | ✗ | ✗ | ✗ |
| PointRCNN [23] | ✗ | ✗ | ✓ | ✗ | ✗ | ✗ |
| Fast-Point RCNN [24] | ✗ | ✗ | ✓ | ✗ | ✗ | ✗ |
| Part-A2 [25] | ✓ | ✗ | ✓ | ✗ | ✗ | ✗ |
| Voxel R-CNN [26] | ✗ | ✗ | ✓ | ✗ | ✗ | ✗ |
| Voxel Transformer [27] | ✗ | ✗ | ✓ | ✗ | ✗ | ✗ |
| Focals Conv [28] | ✗ | ✗ | ✓ | ✗ | ✗ | ✗ |
| BEV-Fusion [29] | ✓ | ✗ | ✓ | ✗ | ✗ | ✗ |
| TensorRT [30] | ✗ | ✗ | ✗ | ✓ | ✓ | ✓ |
| XLA [31] | ✗ | ✗ | ✗ | ✓ | ✓ | ✓ |
| ONNX [7] | ✗ | ✗ | ✗ | ✓ | ✓ | ✓ |
| TVM [32] | ✗ | ✗ | ✗ | ✓ | ✓ | ✗ |
| EMOS (**Ours**) | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |

2.6 times faster on NVIDIA Jetson. On the nuScenes, it surpasses the SOTA with gains of 0.2% mAP and 0.5% NDS, while using only 66% of the model size and achieving a 15.35 times speedup on Jetson. These results clearly highlight the dual advantages of *EMOS* in accuracy and efficiency, making it well-suited for real-time autonomous driving.

## II. RELATED WORK

Improving both efficiency and accuracy in ML systems is a highly challenging task that requires a careful balance of optimizations across multiple levels. Achieving overall performance improvements demands not only algorithmic advances but also system-level enhancements. We provide a systematic comparison of current state-of-the-art approaches at both levels, with detailed results summarized in Table II, offering a clear perspective on diverse optimization strategies. The remainder of this chapter presents an in-depth discussion of related work on object 3D detection algorithms, ML compilation platforms, and MoE frameworks.

### A. 3D Object Detection Algorithms

*Camera-only Algorithms:* Current ADS widely adopt monocular and stereo cameras as imaging sensors. For monocular cameras, research divides into three main categories [33]: (1) Prior-guided methods [34], [35], [36], (2) Camera-only approaches [37], [38], [39], and (3) Depth-assisted techniques [40], [41], [42]. For stereo cameras, research falls into two categories: (1) 2D-detection-based methods [43], [44], [45] and (2) Volume-based approaches [46], [47], [48]. The main advantage of camera-based methods is their ability to capture detailed image information along object edges, aiding effective object segmentation. However, cameras cannot provide real-time distance measurements for every point, leading to inferior performance in 3D object detection compared to LiDAR. Processing dense image signals also demands substantial computational cost. Consequently, camera-based methods generally underperform in efficiency compared to LiDAR-only methods in 3D object detection.

*LiDAR-only Algorithms:* Owing to its active 3D signal generation, LiDAR has been widely adopted in ADS for 3D object detection [23], [24], [49], [50], [51], [52]. Some approaches predict 3D bounding boxes from point clouds by projecting them into voxels [22], [53] or pillars [51], whereas others directly use raw point clouds as input [54], [55]. Anchor-based detection heads are commonly employed [22], [51], although center-based representations have also been explored [56], [57]. For LiDAR-only methods, accuracy on KITTI car detection has substantially improved: 3DSSD [55], PartA2 [50], and Voxel R-CNN [26] all achieve over 85% on the *easy* split, but performance drops to around 72% on the *hard* split, indicating limited robustness

under challenging conditions. However, due to safety regulations, LiDAR sensors are subject to strict limitations on signal intensity. This constraint leads to weaker signal reflections when interacting with surfaces at small incidence angles or with soft materials such as skin and cloth. Consequently, LiDAR-based methods tend to exhibit reduced accuracy for the "person" class on the KITTI dataset.

*Multimodal Fusion Algorithms:* Although LiDAR provides accurate geometric measurements, the absence of RGB information limits its effectiveness in detecting small objects and appearance-dependent categories such as traffic signs. Consequently, fusing LiDAR and camera data has attracted extensive research attention in ADS. To align LiDAR and image features across different dimensions and coordinate systems, two primary approaches have been proposed [33]: projection-based and model-based alignment. Projection-based methods either map image features onto raw point clouds to preserve the original representation [58], [59], [60], [61], or combine point cloud and image features during feature extraction [28], [62], [63], [64]. In contrast, model-based methods align multimodal features via query-based cross-attention before fusion [29], [65], [66], [67], or employ a unified feature space for joint representation [27], [68], [69], [70]. On the KITTI car detection benchmark, multimodal fusion methods have reported strong performance. PPF-Det achieves 89.51, 84.46, and 78.91 AP on the *easy*, *moderate*, and *hard* splits, respectively [71]; RoboFusion reports 91.75, 84.08, and 80.71 [72]; and GraphAlign++ attains 90.98, 83.76, and 80.16 [73].

### B. Machine Learning Compilation Platforms

Advanced ML compilers and deployment tools are essential for the effective utilization of a wide range of deep learning acceleration hardware. With the increasing complexity and computational demands of ML models, specialized deployment frameworks allow TensorFlow and PyTorch models to achieve efficient execution across diverse hardware architectures. ML compilation platforms can be categorized into three groups: general-purpose, hardware-specific, and framework-specific platforms.

*General-Purpose Compilation Platforms:* ONNX Runtime [74] is an open standard that supports model interoperability and enables portable execution across CPUs, GPUs, and specialized accelerators. It improves performance through graph-level and memory management optimizations. Multi-level Intermediate Representation (MLIR) [75], built on LLVM [76], a widely used modular compiler infrastructure, standardizes ML compiler pipelines and enhances interoperability and scalability across frameworks and hardware platforms. Apache TVM [32] is an open-source deep learning compiler that supports automated tuning and tensor scheduling for efficient deployment on heterogeneous hardware, including CPUs, GPUs, and FPGAs. It has been widely applied in both cloud-based and edge AI scenarios for inference and training.

*Hardware-Specific Compilation Platforms:* TensorRT [30], developed by NVIDIA, accelerates deep learning inference on NVIDIA GPUs through quantization, operator fusion, and layer optimizations, enabling high throughput and low latency, and is widely applied in real-time inference scenarios. OpenVINO [77], Intel's optimization toolkit, optimizes deep learning inference performance on Intel CPUs, GPUs, and FPGAs through computation graph optimizations and model quantization. OneDNN [78] provides optimized deep learning primitives by employing CPU vectorization and memory optimizations to support efficient inference execution.

*Framework-Specific Compilation Platforms:* Glow [79], an open-source compiler from Meta, optimizes PyTorch models through ahead-of-time compilation and quantization, and supports efficient execution on CPUs, GPUs, and accelerators, and is commonly applied in mobile and embedded environments. TorchDynamo [80], introduced in PyTorch 2.0, is a Just-in-time (JIT) compiler that mitigates Python execution overhead and integrates with backends such as XLA [31], Inductor, and TensorRT to improve model execution performance.

## III. Method

The pipeline of *EMOS* is illustrated as Fig. 2. Upon receiving input data, the Adaptive Multimodal Expert Router (Section III-D) firstly preprocesses the data and selects the most suitable expert from the Expert Candidate Pool (Section III-A) and applies our designed MoE ensemble strategy (Section III-C) to get the final detection result. Due to the challenging training phase of MoE framwork, we propose a hierarchical training strategy for *EMOS*, which is detailedly illustrated in Section III-E.

### A. Expert Utility Evaluation

Various approaches have been proposed for 3D object detection [51], [53], [54], [57], differing in model scale, efficiency, and input modalities such as LiDAR, camera, radar, or multimodal combinations [29], [50], [81]. This heterogeneity makes apples-to-apples comparisons difficult. To provide a fair and unified lens, we introduce two novel utility functions to profile the existing experts: the *Accuracy Utility Function* $\mathcal{U}_a$ and the *Efficiency Utility Function* $\mathcal{U}_e$.

*The intuition is simple: good models do more with less:* $\mathcal{U}_a$ measures how effectively a model converts capacity into predictive quality – rewarding accuracy gains per unit of capacity (e.g., per million parameters), so models that extract more information from the same parameter budget score higher. $\mathcal{U}_e$ measures the accuracy delivered per unit of execution cost on a target platform (e.g., per millisecond of end-to-end latency or per joule), aligning comparison with deployment realities where latency and energy typically grow with model size and input resolution on fixed hardware. Together, $(\mathcal{U}_a, \mathcal{U}_e)$ enable principled, modality-agnostic comparisons across architectures under consistent task and hardware constraints. They can be used as a standardized profiling protocol that accounts jointly for accuracy, model size, input modality, and inference cost.

*Accuracy Utility Function:* Let $\mathcal{A}_{m,d}$ denote the accuracy (e.g., mAP or NDS) of model $m$ on dataset $d$, $\mathcal{P}_{m,d}$ its parameter count, and $\mathcal{D}_{m,d}$ the effective input information volume, which scales with sensor resolution, frame rate, and the number/type

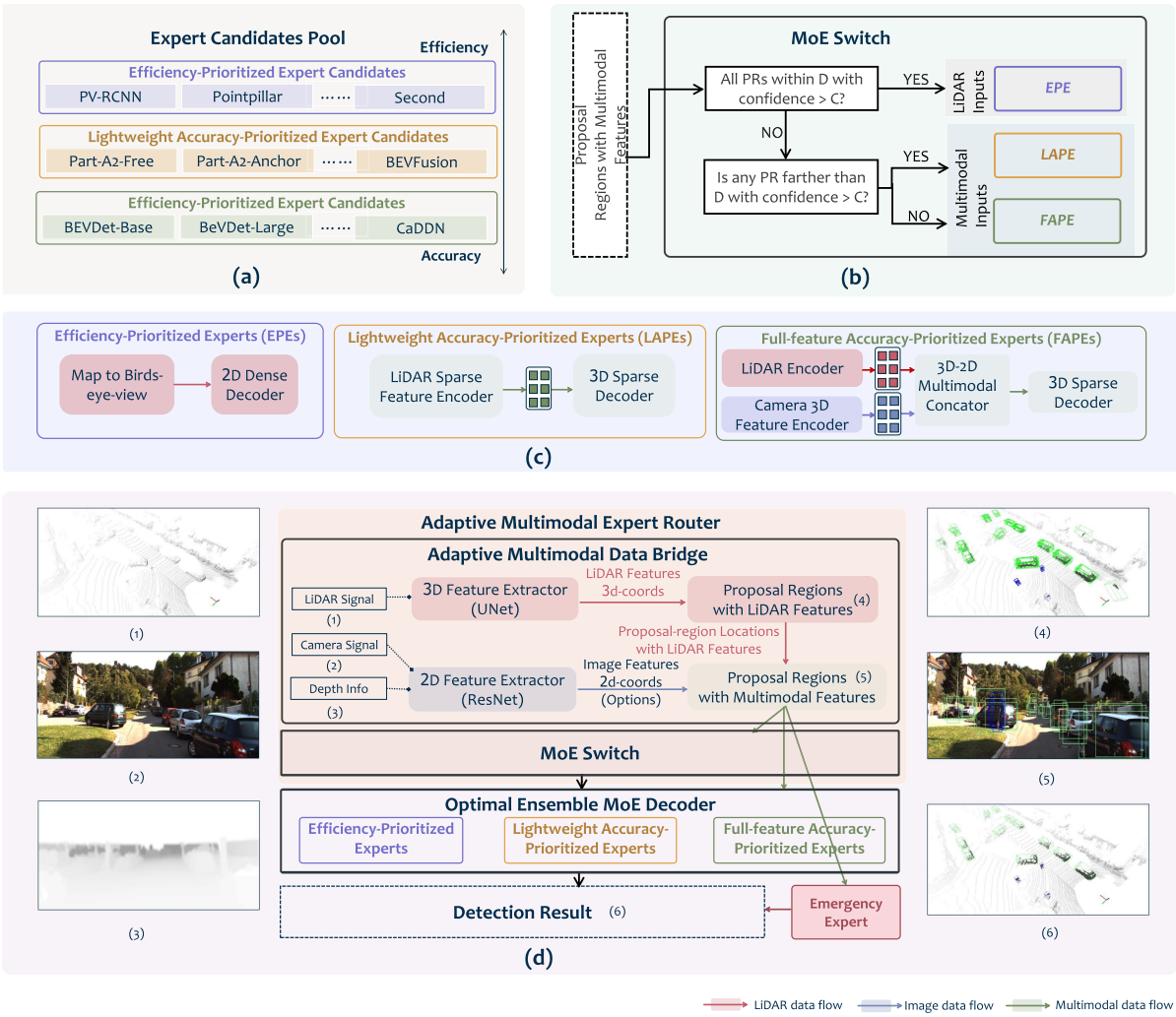



Fig. 2. Overall Framework and Submodules of *EMOS*. (a) The *Expert Candidates Pool*, which organizes expert models into different categories according to efficiency–accuracy utility trade-offs. (b) The *MoE Switch*, which determines expert selection based on proposal-region confidence and distance. (c) The *Optimal Ensemble MoE Decoder* serves as the second-stage decoder for information post-processing. It incorporates three categories of experts: *Efficiency-Prioritized Experts (EPEs)*, *Lightweight Accuracy-Prioritized Experts (LAPEs)*, and *Full-feature Accuracy-Prioritized Experts (FAPEs)*, each with distinct internal structures. (d) The end-to-end pipeline of *EMOS*, illustrating each processing stage together with representative visual results.

of sensing modalities. To disentangle accuracy gains attributable to input data richness from those due to model architecture and capacity, we define two utilities using model setting normalizers utility $\mathcal{U}_{\text{data}}$ and the model design and parameters settings $\mathcal{U}_{\text{param}}$:

$$\mathcal{U}_{\text{data}}(m,d) = \frac{\mathcal{A}_{m,d}}{\delta \log\left(\mathcal{D}_{m,d}\right) + b}, \tag{1}$$

$$\mathcal{U}_{\text{param}}(m,d) = \frac{\mathcal{A}_{m,d}}{\gamma \log\left(\mathcal{P}_{m,d}\right) + c}, \tag{2}$$

where $\gamma$ and $\delta$ control the sensitivity to parameter size and input data resolution, respectively, and $c$ and $b$ are stabilizing bias terms. Both terms reflect the principle that accuracy improvements become increasingly costly as models grow larger or sensor resolutions increase, following a logarithmic penalty. The overall $\mathcal{U}_a$ is:

$$\mathcal{U}_a(m,d) = \alpha \mathcal{U}_{\text{param}}(m,d) + \beta \mathcal{U}_{\text{data}}(m,d), \tag{3}$$

where $\alpha$ and $\beta$ are balancing weights that determine the relative contribution of parameter and data efficiencies.

*Efficiency Utility Function:* Let $\mathcal{T}_{m,d}$ denote the inference time of the model $m$ on dataset $d$, and $\mathcal{P}_{m,d}$ denote its parameter size. To capture the diminishing efficiency of large models, we apply a negative logarithmic penalty to parameter size, and complement it with a latency–size ratio term that reflects hardware utilization. The overall efficiency score is defined as:

$$\mathcal{U}_e(m,d) = -\mathcal{H} \log\left(\mathcal{P}_{m,d}\right) + \frac{\nu_h \mathcal{T}_{m,d}}{\theta \mathcal{P}_{m,d}} + C, \tag{4}$$

where $\mathcal{H}$ scales the logarithmic size penalty, $\theta$ adjusts the sensitivity of the size–latency ratio, $\nu_h$ is a hardware-dependent coefficient that adapts the latency penalty across different deployment platforms, and $C$ is a constant bias term. The inclusion of $-\log(\mathcal{P}_{m,d})$ ensures that excessively large models are penalized sharply, which is consistent with empirical observations

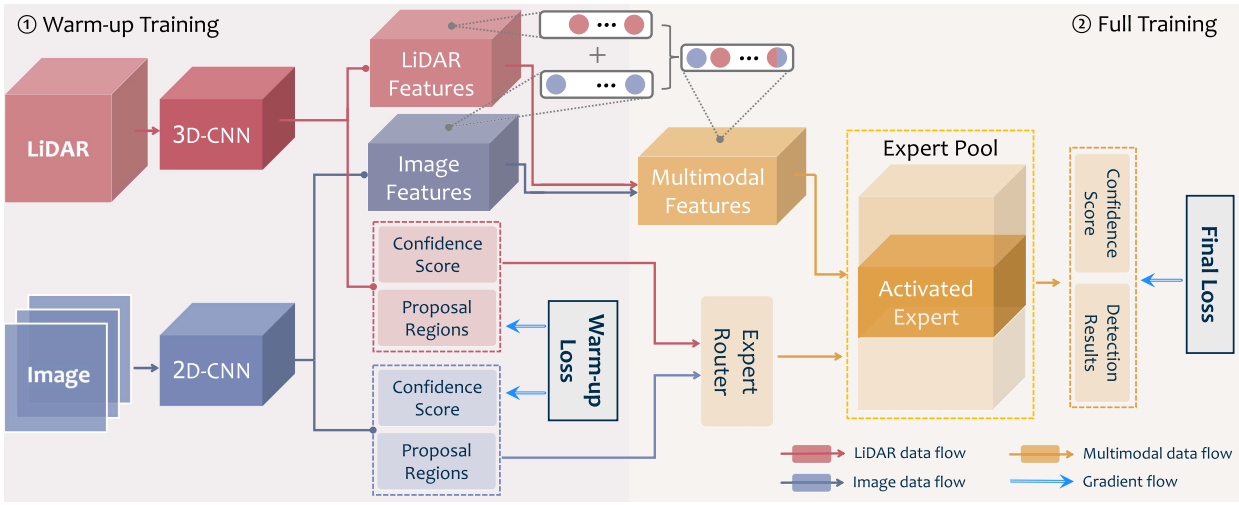



Fig. 3. Multimodal Data Flow and Hierarchical Training Design of *EMOS*. The three paths include backpropagation from: (1) Final detection output, (2) LiDAR branch of *AMDB*, and (3) Image branch of *AMDB*. Since expert training heavily depends on the outputs of these two branches, which are unreliable in early epochs, introducing additional backpropagation paths directly supervising the two branches of *AMDB* accelerates training and improves convergence quality.

that model size often increases much faster than the associated accuracy gains.

By jointly applying $\mathcal{U}_a$ and $\mathcal{U}_e$, candidate methods for 3D object detection can be consistently evaluated across datasets and hardware platform. Models with high $\mathcal{U}_a$ represent *Accuracy-Prioritized Experts (APEs)*, which excel in complex conditions but incur higher cost, while models with high $\mathcal{U}_e$ correspond to *Efficiency-Prioritized Experts (EPEs)*, which are lightweight and suitable for simple environments. In the following sections, we integrate both groups into our MoE framework and demonstrate how adaptive expert selection balances accuracy and efficiency on resource-constrained platforms.

## B. Expert Profiling

*Efficiency-Prioritized Experts: EPEs* are designed for fast inference speed. They first project 3D LiDAR features extracted by the Adaptive Multimodal Data Bridge (*AMDB*) into Bird's Eye View (BEV) representation. In scenarios with abundant voxel information, accurate height cues are preserved even after compression into BEV space. To efficiently process near-field objects with dense and well-defined voxel representations, *EPEs* use 2D decoder to extract features and perform localization and classification, then employ a transformation module to map the 2D results back into 3D space, prioritizing inference speed while maintaining reliable detection accuracy.

*Accuracy-Prioritized Experts (APEs), including two types: LAPEs and FAPEs. APEs* are designed for challenging scenarios such as far-field objects and incomplete voxel information. Speicfically, *LAPEs* project both 3D LiDAR and image features extracted by *AMDB* into a BEV representation, followed by a trained decoder that predicts per-voxel classifications and bounding boxes. On the other hand, *FAPEs* adopt a more comprehensive framework that except 3D LiDAR and image features extracted by *AMDB*, it adds two separate encoders to further

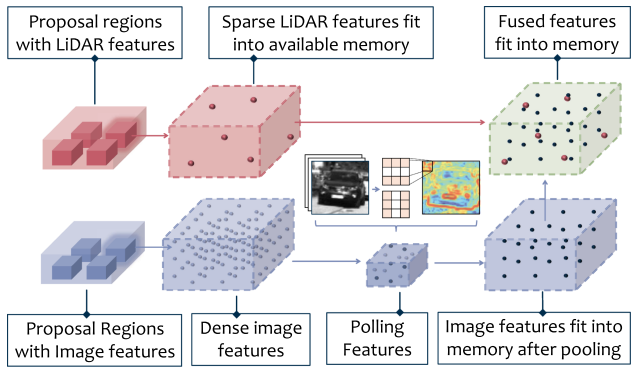



Fig. 4. Demonstration of Multiscale Pooling and Multimodal Fusion. Image features are first aligned with the 3D proposal space generated by the LiDAR branch of AMDB. Relevant image regions are extracted, projected into 3D space, and subsequently processed through pooling and fusion. After pooling, image features whose pixels correspond to the same 3D voxel are concatenated with the associated LiDAR voxel features, whereas unmatched pixels are concatenated with zero-padded features, details are illustrated in Algorithm 1.

encode the extracted LiDAR and image features, respectively. The extracted features are then fused through a multimodal decoder. By leveraging both LiDAR and image modalities, *APEs* effectively mitigate the issue of missing LiDAR details. To further enhance efficiency, *APEs* integrate a multiscale pooling module that reduces redundant computation before performing multimodal fusion of image and LiDAR voxel features (see Algorithm 1 and Fig. 4). This design improves detection performance, particularly for distant and challenging objects.

*Emergency Expert:* The *Emergency expert* designed for situations where the full perception pipeline cannot be executed in time (e.g., the vehicle is moving too fast or an obstacle is dangerously close). To ensure timely reaction, it directly consumes raw sensor data without preprocessing. We define an emergence score $S_E(v, d)$ based on the vehicle speed $v$ and the

**Algorithm 1:** Fusion of Image Features into 3D Space.

**Require:** Features and 3D coordinates of LiDAR voxels $\mathcal{V} = \{f_{v_i}, v_i\}_{i=1}^{N}$, Features and 3D coordinates of image pixels $\mathcal{P} = \{f_{p_j}, p_j\}_{j=1}^{M}$, Proposal regions derived from LiDAR data $\mathcal{R} = \{r_k\}_{k=1}^{K}$

**Ensure:** Fused multimodal features $\mathcal{V}^{\text{fused}}$

Initialize $\mathcal{V}^{\text{fused}} = \{v_i^{\text{fused}} \mid v_i^{\text{fused}} = [f_{v_i}, 0], \forall v_i \in \mathcal{V}\}$
**for** $r_k \in \mathcal{R}$ **do**
  Identify pixels $\mathcal{P}_k = \{p_j \mid p_j \in r_k\}$
  **for** $p_j \in \mathcal{P}_k$ **do**
    **if** $p_j$ exists in $\mathcal{V}$ **then**
      $v_i^{\text{fused}} \leftarrow [f_{v_i}, f_{p_j}]$
    **else**
      $\mathcal{V}^{\text{fused}} \leftarrow \mathcal{V}^{\text{fused}} \cup \{[0, f_{p_j}]\}$
    **end if**
  **end for**
**end for**

distance to the nearest object $d$:

$$S_E(v, d) = \frac{v}{d} \tag{5}$$

When the score exceeds a safety threshold $\tau$, the *Emergency expert* will be activated and immediately issues a braking signal. This formulation naturally captures the intuition that higher speeds and shorter distances both increase collision risk.

### C. MoE Ensemble Strategy

According to (3) and (4), there exists a clear boundary between efficiency and accuracy in the $\mathcal{U}_a$ and $\mathcal{U}_e$. When the majority of errors are concentrated on objects with higher recognition difficulty, further accuracy gains typically demand a substantial increase in model size. In such cases, achieving a marginal gain in accuracy often corresponds to approximately doubling the parameter count, reflecting the diminishing returns of scaling.

To overcome this boundary and jointly optimize efficiency and accuracy, we propose an ensemble learning framework. The key idea is to dynamically assign samples with varying difficulty to the most appropriate expert, rather than always relying on one single end-to-end detection model. Formally, we define the scene-assignment function $\pi(s)$ as:

$$\pi(s) = \begin{cases} EPEs, & \text{if } \hat{\mathcal{F}}(s) < \tau, \\ APEs, & \text{if } \hat{\mathcal{F}}(s) \geq \tau, \end{cases} \tag{6}$$

$\hat{\mathcal{F}}(s)$ denotes the estimated difficulty of scene $s$ and $\tau$ is a threshold. In real-world applications, $\hat{\mathcal{F}}(s)$ is typically defined using a combination of scene-level metrics computed from object-detection outputs (e.g., confidence scores, mAP). Intuitively, scenes with lower difficulty are routed to *EPEs*, whereas more complex cases are handled by *APEs*. This strategy leverages the complementary statistical biases of the two model types to break the efficiency–accuracy bound in (3) and (4). Details of the difficulty modeling and assignment mechanism are provided in Section III-D.

### D. Adaptive Multimodal Expert Router (AMER)

To seamlessly integrate *EPEs* and *APEs* into a unified MoE system, we propose an adaptive multimodal expert router that dynamically selects the most suitable expert for each task. This router comprises two key modules: the Adaptive Multimodal Data Bridge (*AMDB*) and the *MoE Switch*.

*Adaptive Multimodal Data Bridge (AMDB):* LiDAR provides accurate 3D spatial information but often fails to capture low-reflectivity regions, leading to sparse sampling. Images, in contrast, contain rich visual detail but introduce high redundancy, increasing computational cost. To balance these trade-offs, we propose *AMDB*, which leverages LiDAR data for preliminary image segmentation and augments selected voxels in those regions with image features. This design supplies sufficient multimodal information for downstream experts while avoiding excessive computation. The LiDAR branch adopts a voxel-based backbone to extract LiDAR features. Next, fully connected layers generate proposal regions and associated confidence scores, which are subsequently delivered to the proposal regions with the multimodal feature block, guiding the image feature selection, as shown in Fig. 4. The image feature branch, in turn, employs a 2D convolutional network with depth-guided projection into 3D space.

*Scenario-aware Experts Switch:* To balance efficiency and accuracy across diverse driving scenarios, and to better utilize the candidate experts, we introduce the *MoE Switch* module. This component dynamically assigns each scenario to the most suitable expert based on scene-specific latency and accuracy requirements, as illustrated in Fig. 2(b). By training experts specialized for distinct conditions and routing each scenario accordingly, the *MoE Switch* reduces average inference time while maintaining accuracy within safety thresholds. During inference, expert selection is guided by object distance and clarity, where clarity is inferred from the confidence scores of proposal regions generated by the *AMDB*. Intuitively, *EPEs* are activated when all objects are clearly visible and no distant objects are present, while *LAPEs* are selected for scenes with unclear objects but no distant ones, or with distant objects that remain clear and *FAPEs* are reserved for scenarios containing both unclear and distant objects.

### E. EMOS Hierarchical Training Strategy

*Hierarchical Multimodal Loss and Back-Propagation:* During training, experts heavily depend on proposal regions generated by the *AMDB*. However, in early epochs these proposals are often unreliable, which disrupts the decision-making of the *MoE Switch* and introduces noise into expert training. To address this, we design a hierarchical back-propagation strategy (Fig. 3), comprising three supervision routes: (1) proposal regions and confidence scores from the LiDAR branch of the *AMDB*; (2) proposal regions and confidence scores from the Image branch of the *AMDB*; and (3) final detection outputs and confidence scores from each expert. In addition, we pretrain both branches of the *AMDB* and the experts independently to stabilize their outputs prior to joint training. Each route is supervised by a dedicated loss function, which together constitute our hierarchical

multimodal loss:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{cls}} + \frac{\mathcal{L}_{\text{reg}}}{N_{\text{fore}}} \tag{7}$$

where $N_{\text{fore}}$ represents foreground voxels associated with objects to be detected. The classification loss $\mathcal{L}_{\text{cls}}$ and regression loss $\mathcal{L}_{\text{reg}}$ are formulated as:

$$\mathcal{L}_{\text{cls}} = \mathcal{L}_{\text{ce}}(y_c, \hat{y}_c) \tag{8}$$

$$\mathcal{L}_{\text{reg}} = \mathcal{L}_{\text{smooth-L1}}(y_r, \hat{y}_r) \tag{9}$$

where $\hat{y}_c$ and $\hat{y}_r$ denote the predicted class and location of proposal regions (for *AMDB* supervision) or final detections (for expert supervision), and $y_c$ and $y_r$ are their corresponding ground-truth (GT) values. $\mathcal{L}_{\text{ce}}$ represents the binary cross-entropy loss for classification, while $\mathcal{L}_{\text{smooth-L1}}$ denotes the smooth-L1 loss for localization. This hierarchical training strategy not only stabilizes expert learning but also improves the model's adaptability across different data granularities. Consequently, it enables effective use of multiscale pooling for memory efficiency, even under compressed data representations, as discussed in Section IV-A.

*Addressing Long-tail Effects in MoE Training:* In traditional MoE frameworks [82], long-tail effects arise when certain experts receive insufficient training due to imbalanced data distributions or skewed allocation policies [83], which results in reduced performance and inefficient resource utilization. In datasets such as KITTI and nuScenes, the limited number of training samples further exacerbates this issue: some experts, particularly *EPEs*, exhibit reduced performance on their designated scenarios when disproportionately exposed to samples from other scenarios. To address this, we propose two strategies. First, during the fine-tuning stage, we introduce a data subset division scheme that partitions training data into $K$ subsets $C_1, C_2, \ldots, C_K$, each assigned to an expert $E_i$. Each subset $C_i$ consists of a disjoint target set $\mathcal{T}_i$ and a shared interference set $\mathcal{I}_i$, with $\mathcal{T}_i \cap \mathcal{I}_i = \varnothing$ for all $i$. Second, during pretraining, we increase the sampling frequency of subsets with fewer samples, ensuring that each class is exposed to the model an equal number of times. This re-balancing strategy prevents experts assigned to rare scenarios from being insufficiently trained. Finally, to mitigate interference within each subset, we incorporate regularization terms into expert loss functions, promoting attention to target classes while suppressing misleading gradients from interference:

$$\mathcal{L}_{\text{exp-cls}} = \mathcal{L}_{\text{ce}}(y_c, \hat{y}_c) + \mathcal{R}_{\text{cls}} \tag{10}$$

$$\mathcal{L}_{\text{exp-reg}} = \mathcal{L}_{\text{smooth-L1}}(y_r, \hat{y}_r) + \mathcal{R}_{\text{reg}} \tag{11}$$

where $\mathcal{L}_{\text{exp-cls}}$ and $\mathcal{L}_{\text{exp-reg}}$ denote the classification and regression losses for final detections, respectively. The regularization terms are then formulated as:

$$\mathcal{R}_{\text{cls}} = -\alpha(1 - p_t)^{\gamma} \log(p_t) \tag{12}$$

$$\mathcal{R}_{\text{reg}} = \sum_i \lambda \left(\hat{y}_i - y_i\right)^2 \tag{13}$$

where $p_t$ is the probability of interference presence, $\gamma$ emphasizes hard samples, $\alpha$ balances sample contributions, and $\lambda$ is a weighting hyperparameter. The settings of parameters are illustrated in Section V-B.

## IV. Computing Systems Optimization

Beyond the core design and training algorithms of our MoE framework, efficient edge deployment remains a key challenge. Accordingly, we proposed: algorithmic optimizations that reduce computation and memory footprint (Section IV-A), and hardware–software co-optimization tailored to the target platforms (Section IV-B).

### A. Algorithmic Efficiency Improvement on Edge

Edge devices operate under stringent resource constraints, manifested as higher CPU–GPU transfer latency, limited physical memory, and incomplete library support, which underscores the need for careful memory management and operator-level optimization. In our pipeline, convolutional operators, particularly 3D sparse convolutions, dominate inference time. To mitigate this bottleneck, we employ an edge-optimized 3D sparse convolution scheme [84] and introduce a multiscale pooling module that strengthens multimodal feature fusion while simultaneously reducing computational and memory overhead.

*System Innovation 1. Edge-optimized 3D Sparse Convolution:* Our proposed 3D sparse convolution reduces redundant computation by focusing on nonzero voxels, yielding 65–80% fewer operations compared with dense convolution [53]. To further align with hardware constraints on edge devices, we design a hardware-aware sparse convolution library, featuring parallelized kernels and fused FMA operations, seamlessly integrated into the ONNX Runtime compiler. The system-level optimization methods are presented in Section IV-B.

*System Innovation 2. Multiscale Pooling:* When processing multimodal inputs, the combined voxelized features from LiDAR and images frequently exceed the cache capacity of edge devices, thereby limiting computational efficiency. To address this issue, we introduce a multiscale pooling strategy, as illustrated in Fig. 4, which adaptively downsamples image feature voxels to fit cache constraints and subsequently fuses them with LiDAR voxels, enabling both computational efficiency and effective multimodal fusion. This design achieves both efficient computation and effective multimodal integration. Voxel selection is guided by an adaptive filter based on the Sobel operator, which prioritizes informative regions. The pooling size is user-configurable within the ONNX framework, providing flexibility to accommodate different hardware constraints.

### B. Optimizing Hardware & Software for MoE System

To further increase *EMOS*'s efficiency, at the system level, we implement memory and computation graph optimization to streamline execution, ensuring efficient inference across diverse hardware platforms.

*System Innovation 3. Memory Optimization:* We propose four strategies to accelerate the runtime of memory usage, including:

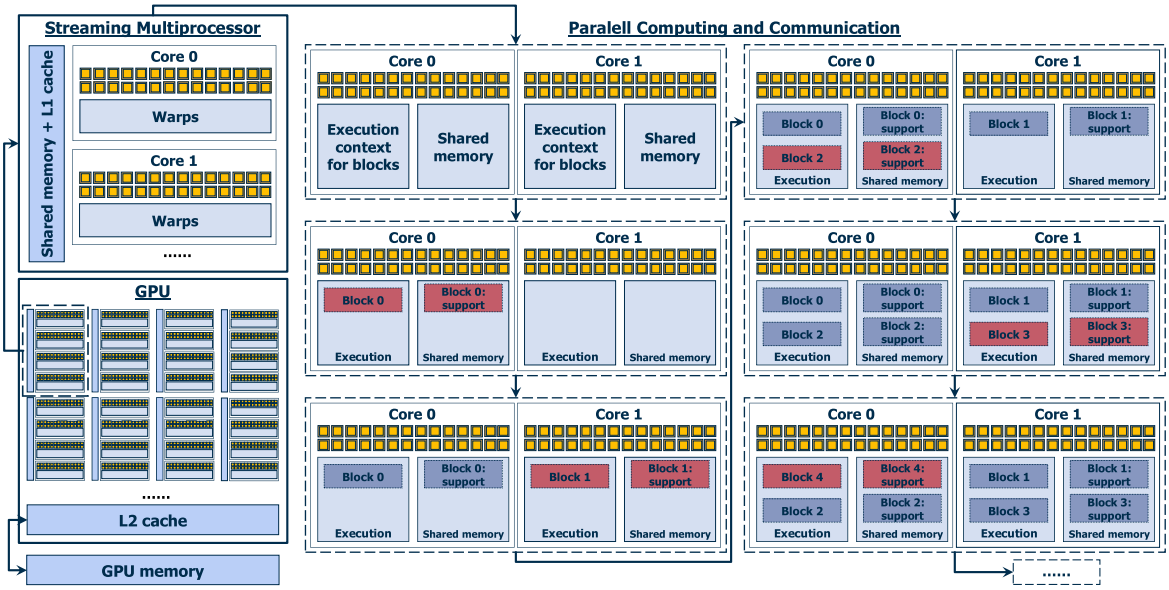



Fig. 5. Illustration of Streaming Multiprocessors (SMs), Shared Memory, and Parallel Operations. SMs form the core computational units of a GPU, each containing cores for parallel execution. When a kernel is launched, it defines computation blocks that will be distributed across SMs. Shared memory provides high-speed memory support for blocks within an SM, facilitating data exchange and reducing memory latency. Parallel Communication and Computation operate as follows: (1) The host dispatches a kernel to the GPU, defining computational task; (2) The scheduler assigns block 0 to an SM, reserving execution contexts and shared memory; (3) Additional blocks are mapped to available execution contexts in an interleaved manner; (4) Once block 0 completes, block 4 is assigned to the freed core, and this process iterates.

**Algorithm 2:** Optimized Edge-Based Sparse 3D Convolution.

**Require:** Input non-empty voxels $\mathcal{V}_{\text{in}} = \{f_{\text{in}}^i, o_{\text{in}}^i\}_{i=1}^N$, where $f$ represents input features and $o$ represents input voxel coordinates; Convolution kernel $\mathcal{K}$; Voxel Space $\mathcal{H}$.

**Ensure:** Output non-empty voxels $\mathcal{V}_{\text{out}} = \{f_{\text{out}}^i, o_{\text{out}}^i\}_{i=1}^N$.

1: **for** $v_{\text{in}}^i = \{f_{\text{in}}^i, o_{\text{in}}^i\} \in \mathcal{V}_{\text{in}}$ **do in parallel**

2: $(\mathrm{x}_{k_0}, \mathrm{y}_{k_0}, \mathrm{z}_{k_0}) \leftarrow o_{\text{in}}^i$;*# Align kernel center to voxel*

4: *# Collect all voxels currently covered by kernel*

5: $\mathcal{V}_{\text{part}} \leftarrow \{(f_{\text{in}}^j, o_{\text{in}}^j) \mid o_{\text{in}}^j = (\mathrm{x}_c + \mathrm{kx}, \mathrm{y}_c + \mathrm{ky}, \mathrm{z}_c + \mathrm{kz}) \; \forall(\mathrm{kx}, \mathrm{ky}, \mathrm{kz}) \in \mathcal{K}, o_{\text{in}}^j \in \mathcal{V}_{\text{in}}, o_{\text{in}}^j \in \mathcal{H}\}$

7: *# Perform Fused Multiply-Add (FMA) as convolution*

8: $(f_{\text{out}}^i, o_{\text{out}}^i) \leftarrow \mathbf{FMA}(\mathcal{V}_{\text{part}}, \mathcal{K})$

9: **end for**

1) *Reducing global memory access [85]:* Shared memory, illustrated in Fig. 5, serves as a high-speed cache within streaming multiprocessors (SMs), enabling efficient data sharing among cores. Leveraging shared memory notably improves GPU throughput and reduces latency. In this work, we configure the ONNX compiler to maximize shared-memory hit rates and optimize warp-level execution order, thereby eliminating redundant memory access and improving overall computational performance.

2) *Parallel Communication and Computation [86]:* In baseline PyTorch pipelines, the common practice is to transfer the entire batch to the GPU before computation starts, which effectively serializes communication and computation. To fully utilize GPU compute cores and interconnect bandwidth, we partition tensors into multiple chunks. While chunk $i$ is being processed, chunk $i+1$ is prefetched into device memory and chunk $i-1$ is offloaded, thereby overlapping communication with computation and improving end-to-end throughput (see Fig. 5).

3) *Thread Management [87]:* To enhance memory efficiency, we propose a four-stage thread management strategy. At the end of each stage, the CPU terminates all completed threads, issues a system interrupt, and initiates a new batch of threads for the subsequent stage. This staged design ensures that, at any given point, memory retains only the minimal thread-related information required for the current stage. Consequently, effective memory utilization is optimized at the system level. The detailed management process is illustrated in Fig. 6.

4) *Prefix scanning for sparse CNN operation [88]:* To further optimize hardware efficiency, we leverage GPU-based prefix scan capabilities. This parallel technique (often used to compute cumulative sums or products) reduces the number of calculation time steps from $O(N)$ to $O(\log N)$, improving load balancing and overall hardware efficiency. To further enhance hardware performance, we exploit GPU-based prefix scan primitives. This parallel method, commonly employed for computing cumulative sums or products, reduces the computational steps from $O(N)$ to $O(\log N)$, thereby improving load balancing and overall throughput.

*System Innovation 4. Computation Graph Optimization:* Our ONNX computation graph optimization occurs in two stages [7]: 1) Graph rewriting (before partitioning on each streaming multiprocessor subcore) and 2) Complex node fusion (after partitioning).

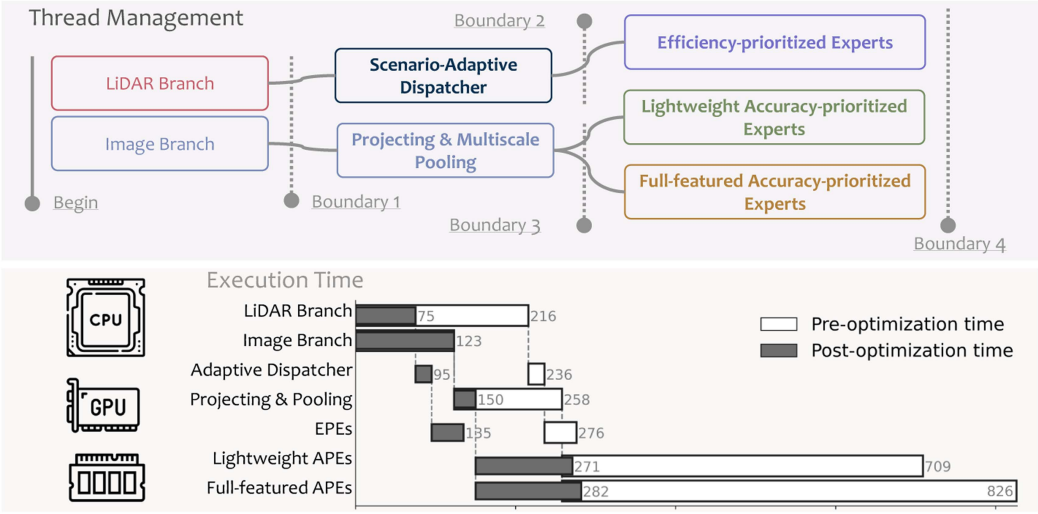



Fig. 6. Illustration of Thread Management. We define four boundaries to manage all threads in *EMOS*. Upon reaching each boundary, the preceding threads are terminated, and their allocated address space is released. Each component spans a set of threads, with light gray strips and numbers indicating execution time before optimization, and darker strips and numbers representing execution time after optimization.

1) *Graph Rewriting:* Graph rewriting encompasses a set of optimization techniques such as constant folding, redundant node elimination, and semantic-preserving operator fusion. Constant folding accelerates computations involving constant values, thereby reducing runtime overhead. Redundant node elimination removes superfluous operations without affecting the computational graph's semantics. Semantic-preserving fusion consolidates multiple operators (e.g., Conv + Add) into a single node, improving execution efficiency.

2) *Complex Node Fusion:* After partitioning, complex node fusion is employed to further improve execution efficiency. In particular, for General Matrix Multiplication (GEMM), we fuse polynomial computations with activation functions to reduce redundant arithmetic operations. Likewise, layer normalization fusion condenses multi-stage procedures (e.g., ReduceMean, Sub, Pow, Sqrt, Div) into a single LayerNormalization operator. Finally, compound expressions such as $\text{LayerNorm}(X + f(X))$ are fused into a unified operation, thereby minimizing intermediate computations and memory accesses.

## V. Experiments

In this section, we evaluate EMOS on two well-established 3D objection datasets: KITTI [10] and nuScenes [2] dataset and evaluating both accuracy and efficiency. To better reflect our proposed method in real-world autonomous driving scenarios, we conduct experiments on both the widely used GPUs (Nvidia RTX A4000, RTX 3080) and edge-computing platform (Nvidia Jetson AGX Orin) (Section V-A). We provide detailed hyperparameters settings in Section V-B and both quantitative and qualitative results of *EMOS* in Section V-C. The ablation study is provided in Section V-D.

### A. Preliminaries

*Datasets:* For evaluation, we train and test *EMOS* on 3D object detection using KITTI [10] and nuScenes [2], two standard benchmarks in autonomous driving research. The KITTI dataset comprises 22 driving scenes with 3,712 training, 3,769 validation, and 7,518 test samples, totaling 80,256 annotated objects. To characterize detection difficulty, objects are categorized as *Easy* (height $\geq$40 px, fully visible, truncation $\leq$15%), *Moderate* (height $\geq$25 px, partly occluded, truncation $\leq$30%), and *Hard* (height $\geq$25 px, heavily occluded, truncation $\leq$50%). These difficulty levels, approximately distributed as 30% *Easy*, 40% *Moderate*, and 30% *Hard*, provide fine-grained evaluation and have been adopted in subsequent benchmarks. The nuScenes dataset contains 1.4 M images and 400 K LiDAR samples, with annotations covering 23 object classes spanning diverse traffic participants. On average, each keyframe includes about seven pedestrians and twenty vehicles, representing dense urban traffic scenarios. Compared to KITTI, nuScenes incorporates multiple sensor modalities, a larger data scale, and scenarios with higher object density and occlusion.

*Evaluation Metrics:* KITTI evaluates performance with the R40 protocol, computing 3D Average Precision (AP) at 40 recall rates using IoU thresholds of 0.7 for cars and 0.5 for pedestrians and cyclists. This metric emphasizes localization accuracy but overlooks other aspects of detection quality. In contrast, nuScenes employs the nuScenes Detection Score (NDS), which integrates mAP with mean true positive errors in location, size, orientation, attributes, and velocity. By normalizing each component to [0,1] and combining them into a single score, NDS provides a more holistic benchmark that reflects both detection accuracy and the precision of estimated box parameters and dynamic states.

Unlike conventional single-model settings, where the size of the python configuration files (`.pth`) directly corresponds to the overall model size, the computation for *EMOS* follows a different principle due to the MoE design. Specifically, the effective parameter size is defined as the sum of a fixed *AMER* block and the weighted contributions of individual experts, expressed as:

$$\mathcal{S}_{\text{total}} = \mathcal{S}_{\text{AMER}} + \sum_{i=1}^{K} \pi_i \, \mathcal{S}_{E_i}, \tag{14}$$

where $\mathcal{S}_{\text{AMER}}$ denotes the parameter size of the *AMER* block, $\mathcal{S}_{E_i}$ the parameter size of expert $E_i$, and $\pi_i$ the activation frequency of expert $E_i$ across scenarios (see Table VII). Under this formulation, the effective parameter count of *EMOS* corresponds to an equivalent `.pth` weight size of approximately 333.3 MB with 32-bit floating-point (FP32) precision.

*Hardware Adaptation:* We test *EMOS* on three distinct hardware platforms: a workstation with an Intel Xeon X86 CPU paired with an NVIDIA RTX A4000 GPU, an NVIDIA RTX 3080 GPU, and an embedded edge device equipped with an ARM CPU and an Amphere GPU (Jetson Orin AGX). Compared to widely used workstation, Jetson is a resource-constrained edge computing platform with only three cache levels and a 4 MB L2 cache per GPU core. Without advanced sparse convolution libraries, matrix operations exceeding this limit incur clear memory swapping overhead, impacting efficiency. All edge-based experiments were conducted on the Jetson Orin platform using the `ONNX-TensorRT` implementation. The software environment was configured via custom shell scripts, which installed the required ONNX, CUDA, TensorRT, and cuDNN dependencies. The TensorRT engine was generated by a customized build script based on the official NVIDIA-AI-IOT interface framework, where the maximum workspace size and precision mode (FP16) were optimized for embedded deployment. During inference, we restricted *EMOS* to four CPU threads, following the thread management strategy in Fig. 6, to ensure consistent CPU utilization across runs. This hardware setup allows efficient execution of 3D object detection models on both server-class and embedded hardware, with balanced utilization of GPU acceleration and controlled CPU concurrency, thereby enabling reproducible latency and throughput measurements in resource-constrained environments.

### B. Implementation Details

*Hyperparameters in Expert Diagnosis:* To implement the utility-based expert evaluation, we set the hyperparameters in (3) and (4) as follows. For the $\mathcal{U}_a$ (3), we use $\gamma = 0.05$, $\delta = 0.05$, $c = 2$, and $b = 2$ to control the logarithmic penalties and stabilizing biases, while $\alpha = 100$ and $\beta = 100$ determine the relative weights of parameter and data efficiencies. For the *Efficiency Utility Function* (4), we use $H = 10$ to scale the size penalty and $\theta = 0.005$ to adjust the sensitivity of the latency–size ratio, and $C = 53$ as a constant bias term. The hardware-dependent coefficient $\nu_h$ is introduced as an empirical scaling factor to approximate Jetson AGX Orin inference latency from desktop GPU measurements. Specifically, we set $\nu_h = 3.25$ when extrapolating from 3080 to Jetson AGX Orin, and $\nu_h = 4.35$ when extrapolating from A4000 to Jetson AGX Orin, based on baseline model latency ratios across representative detection models. With these hyperparameters, we compute expert candidate scores $(\mathcal{U}_a, \mathcal{U}_e)$ for all models on KITTI and nuScenes, enabling a consistent ranking of experts in terms of accuracy–efficiency trade-offs. These rankings form the basis of the expert selection strategy in our MoE framework, ensuring that *APEs* and *EPEs* are assigned according to their strengths under deployment constraints.

*Expert Switch and Ensemble:* We design the dataset-specific expert switching and ensemble strategies for KITTI and nuScenes based on the rankings in Fig. 7, which shows the efficiency–accuracy utility of each expert. To capture detection difficulty for different scenes, we applied a two-sample Kolmogorov–Smirnov (KS) test, which reveals a statistically distributional shift with respect to both distance and confidence score for each sample, yielding distance thresholds of 23.5 m (KITTI) and 35 m (nuScenes). Visibility criteria were further defined by confidence: predictions with scores above 0.65 or below 0.30 (KITTI) and above 0.60 or below 0.30 (nuScenes) were deemed "clearly visible," while intermediate scores were considered "uncertain." These thresholds provide principled decision boundaries for invoking higher-accuracy experts in the second inference stage. Based on these criteria, the evaluation data were partitioned into three subsets: (i) $\mathcal{T}_{S1}$, containing objects within $D$ meters, processed by *EPEs*; (ii) $\mathcal{T}_{S2}$, where objects exceed $D$ meters or are within $D$ but predicted with intermediate scores, assigned to *LAPEs*; and (iii) $\mathcal{T}_{S3}$, where at least one object is both beyond $D$ and predicted with intermediate scores requiring *FAPEs*. This adaptive partitioning ensures that the MoE switching and ensemble mechanism selects the most suitable expert under varying detection difficulties, balancing efficiency and generalization across datasets.

*Backbone Selection in AMER:* The implementation of the *AMER* relies on dataset-specific backbone choices within the proposed *AMDB*. For the LiDAR branch, we adopt a voxel-based backbone: a 3D Sparse CNN is employed for KITTI to extract spatial geometric features, while a Dense-Head Attention block is used for nuScenes to accommodate its larger scale and diverse traffic scenarios. These backbones allow LiDAR features to be extracted efficiently and adapted to the characteristics of each dataset.

In the image feature branch, a 2D convolutional backbone (e.g., CenterNet or ResNet) is used to extract image features, which are subsequently projected into 3D space via depth estimation. The projected image features are then fused with LiDAR proposals generated in the voxel branch, guided by confidence scores predicted by fully connected layers, as shown in Fig. 4. This design ensures that multimodal features are incorporated selectively in regions where they provide complementary information, thereby reducing redundant computation. The overall *AMDB* workflow and backbone selection are illustrated in Fig. 2.

*Hyperparameters in EMOS Training:* For the loss functions defined in (12) and (13), the hyperparameters are adapted to the characteristics of KITTI and nuScenes. Specifically, the focal loss employs a class-balancing factor $\alpha$ with $\alpha_{\text{car}} = 0.55$ and $\alpha_{\text{others}} = 0.45$ to address the imbalance between vehicle and non-vehicle categories. The hardness weighting parameter $\gamma$ is adapted to dataset characteristics and detection difficulty: $\gamma = 0.2$ for KITTI $easy$ cases, $\gamma = 0.8$ for $moderate$ and $hard$ cases, $\gamma = 0.7$ for rare classes in nuScenes (e.g., barrier), and $\gamma = 0.3$ for common classes (e.g., car). The regression–classification trade-off is controlled by $\lambda = 0.45$ to enable stable

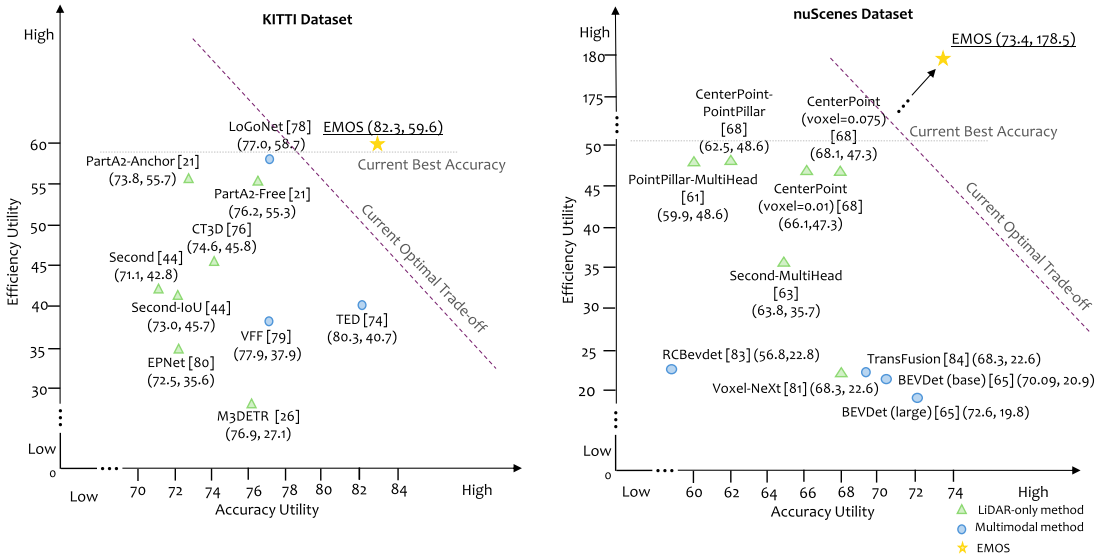



Fig. 7. Accuracy and Efficiency Utility on KITTI and nuScenes. The figures illustrate the distribution of expert models in terms of accuracy utility $\mathcal{U}_a$ and efficiency utility $\mathcal{U}_e$ across two benchmarks. On KITTI, similar trade-offs are observed, with methods such as *PartA2-Free* and *CT3D* favoring efficiency, and *TED* and *VFF* prioritizing accuracy. On nuScenes, LiDAR-only methods (e.g., *CenterPoint*) provide relatively high efficiency with moderate accuracy, while multimodal methods (e.g., *BEVDet*, *TransFusion*) emphasize accuracy but incur higher computational cost. In both datasets, *EMOS* achieves the most balanced trade-off, reaching $(\mathcal{U}_a = 82.3, \mathcal{U}_e = 59.6)$ on KITTI and $(\mathcal{U}_a = 73.4, \mathcal{U}_e = 178.5)$ on nuScenes.

TABLE III
QUANTITATIVE COMPARISON OF DIFFERENT METHODS FOR 3D OBJECT DETECTION ON KITTI. N/A INDICATES THAT EITHER THE OPEN-SOURCE VERSION OF THE RESPECTIVE MODEL DOES NOT SUPPORT THAT PARTICULAR CLASS, OR THE MODEL ITSELF DOES NOT SUPPORT JETSON DEPLOYMENT. DARK BLUE AND LIGHT BLUE MARK THE BEST AND THE SECOND-BEST RESULTS AMONG ALL MODELS.

| Modality | Method | Pedestrian 3D AP (R40) | | | Car 3D AP (R40) | | | Cyclist 3D AP (R40) | | | Inference Time (ms) | | | Model Size (MB) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Easy | Moderate | Hard | Easy | Moderate | Hard | Easy | Moderate | Hard | A4000 | RTX 3080 | Jetson | |
| *LiDAR-only* | M3DETR [73] | 69.69 | 66.04 | 61.61 | 93.96 | 86.28 | 84.32 | 87.88 | 72.47 | 70.63 | 338 | 1157 | N/A | 166 |
| | PV-RCNN [74] | N/A | N/A | N/A | 91.18 | 87.65 | 83.14 | N/A | N/A | N/A | 45 | 207 | 1787 | 50 |
| | Voxel-RCNN [22] | N/A | N/A | N/A | 92.23 | 85.04 | 82.50 | N/A | N/A | N/A | 70 | 240 | N/A | 28 |
| | GLENet-VR [75] | N/A | N/A | N/A | 93.51 | 86.22 | 83.72 | N/A | N/A | N/A | 165 | 575 | N/A | 87 |
| | PartA2-Anchor [21] | 66.87 | 59.68 | 54.60 | 92.15 | 82.92 | 82.10 | 90.34 | 70.05 | 66.89 | 95 | 184 | N/A | 244 |
| | PartA2-Free [21] | 72.31 | 66.36 | 60.06 | 91.66 | 80.28 | 78.08 | 91.88 | 75.33 | 70.67 | 124 | 335 | 1064 | 226 |
| | PointPillar [42] | 57.29 | 51.41 | 46.87 | 87.75 | 78.40 | 75.19 | 81.57 | 62.93 | 58.97 | 41 | 130 | 972 | 18 |
| | Second [44] | 55.94 | 51.14 | 46.16 | 90.55 | 81.60 | 78.60 | 82.96 | 66.73 | 62.78 | 45 | 165 | 1322 | 20 |
| | Second-IoU [44] | 61.10 | 54.66 | 49.50 | 91.53 | 82.36 | 79.62 | 90.73 | 71.23 | 66.26 | 58 | 188 | N/A | 46 |
| | CT3D [76] | 61.05 | 55.57 | 51.10 | 92.34 | 84.97 | 82.91 | 89.01 | 71.88 | 67.91 | 67 | 174 | N/A | 30 |
| *Multimodal* | Voxel-RCNN [22] | N/A | N/A | N/A | 92.08 | 85.90 | 83.36 | N/A | N/A | N/A | 331 | 1057 | 965 | 129 |
| | TED [77] | N/A | N/A | N/A | 92.25 | 88.94 | 86.73 | N/A | N/A | N/A | 100 | 385 | N/A | 65 |
| | LoGoNet [78] | 70.02 | 63.72 | 59.46 | 92.04 | 85.04 | 84.31 | 91.74 | 75.35 | 72.42 | 100 | 342 | N/A | 266 |
| | VFF [79] | 73.26 | 65.11 | 60.03 | 92.24 | 85.51 | 82.92 | 91.74 | 75.35 | 69.84 | 192 | 657 | N/A | 202 |
| | EPNet [80] | 66.74 | 59.29 | 54.82 | 88.76 | 78.65 | 78.32 | 83.88 | 65.60 | 62.70 | 154 | 584 | N/A | 179 |
| | **EMOS (Ours)** | 74.74 | 67.26 | 60.10 | 95.55 | 89.22 | 89.00 | 98.13 | 81.89 | 78.22 | 93 | 295 | 371.3 | 62 |

joint optimization of localization and classification objectives. These hyperparameters define a configuration that emphasizes difficult samples, balances category prevalence, and accommodates the distinct distributions of KITTI and nuScenes.

### C. Results and Analysis

The results of *EMOS* on the KITTI and nuScenes datasets are shown as Tables III and IV, respectively. On both datasets, *EMOS* achieves the best performance in most accuracy-related metrics and efficiency-related metrics. The inference time of *EMOS* on Jetson further highlights *EMOS*'s superiority in edge system adaptation.

*Results on KITTI:* As shown in Table III, *EMOS* improves recognition accuracy by 1.95% (cars), 1.31% (pedestrians), and 6.24% (cyclists). In terms of efficiency, it achieves 371.3 ms per frame on Jetson, providing a 2.6 times speedup over comparable methods.

*Results on nuScenes:* On the nuScenes benchmark (Table IV), *EMOS* improves mAP by 2.8% and NDS by 2.8% compared to efficiency-oriented SOTA models, and further surpasses accuracy-oriented models by 0.2% mAP and 0.5% NDS, while using only 67% of the parameter count. Moreover, it achieves 397.3 ms per frame on Jetson, which is 15.35 times faster than efficiency-oriented baselines, demonstrating strong accuracy–efficiency trade-offs for real-time autonomous driving.

*Results of Accuracy and Efficiency Utility Function:* Fig. 7 presents the joint evaluation of expert models using the proposed $\mathcal{U}_a$ and $\mathcal{U}_e$ on KITTI and nuScenes, respectively. The evaluation shows that models present different trade-offs between accuracy and efficiency: LiDAR-only methods such as CT3D and PartA2-Free obtain intermediate accuracy levels while maintaining higher efficiency compared with other methods, whereas multimodal methods such as TED and BEVDet achieve higher accuracy, accompanied

TABLE IV
QUANTITATIVE COMPARISON OF DIFFERENT METHODS FOR 3D OBJECT DETECTION ON NUSCENES. INFERENCE TIME IS MEASURED BY RUNNING JETSON AGX ORIN PLATFORM. N/A INDICATES THAT EITHER THE OPEN-SOURCE VERSION OF THE RESPECTIVE MODEL DOES NOT SUPPORT THAT PARTICULAR CLASS, OR THE MODEL ITSELF DOES NOT SUPPORT JETSON DEPLOYMENT. DARK BLUE AND LIGHT BLUE MARK THE BEST AND THE SECOND-BEST RESULT AMONG ALL MODELS. FOR INFERENCE TIME, LIGHT BLUE MARKS THE SECOND-BEST MODELS UNDER MULTIMODAL AND OVERALL SETTINGS.

| Modality | Method | Average Precision ↑ | | | | | NDS ↑ | Inference Time (ms) ↓ | Model Size (MB) |
|---|---|---|---|---|---|---|---|---|---|
| | | Car | Truck | Trailer | Bus | mAP | | | |
| LiDAR-only | PointPillar-MultiHead [51] | 81.3 | 50.0 | 72.3 | 63.4 | 44.6 | 58.2 | 456.1 | 23.3 |
| | Second-Multihead [53] | 81.6 | 51.7 | 77.6 | 67.0 | 50.6 | 62.3 | 546.3 | 35.2 |
| | CenterPoint-PointPillar [57] | 83.1 | 50.6 | 78.8 | 62.5 | 50.0 | 60.7 | 457.5 | 23.1 |
| | CenterPoint (voxel-size=0.1) [57] | N/A | N/A | N/A | N/A | 56.0 | 64.5 | 535.2 | 34.3 |
| | CenterPoint (voxel-size=0.075) [57] | N/A | N/A | N/A | N/A | 59.2 | 66.5 | 535.1 | 34.2 |
| | VoxelNeXt [96] | 83.9 | 55.5 | 84.6 | 70.5 | 60.5 | 66.6 | N/A | 31.3 |
| Camera-only | Sparse4D-v3 [97] | 78.3 | 50.8 | 71.3 | 51.5 | 63.0 | 69.4 | N/A | 675.0 |
| Multimodal | RCBevdet [98] | N/A | N/A | N/A | N/A | 45.3 | 56.8 | 6094.8 | 330.7 |
| | CMT [69] | 87.2 | 61.5 | 86.9 | 72.4 | 70.3 | 72.9 | N/A | 992.0 |
| | TransFusion [99] | 86.6 | 60.9 | 86.6 | 73.6 | 68.9 | 71.7 | 6172.4 | 333.9 |
| | BEVDet (base) [81] | N/A | N/A | N/A | N/A | 70.0 | 73.4 | 7321.5 | 401.0 |
| | BEVDet (large) [81] | 89.9 | 67.0 | 47.4 | 78.6 | 71.5 | 74.0 | 8306.4 | 496.0 |
| | **EMOS** (ours) | 89.7 | 69.4 | 47.5 | 80.5 | 71.7 | 74.5 | 397.3 | 333.3 |

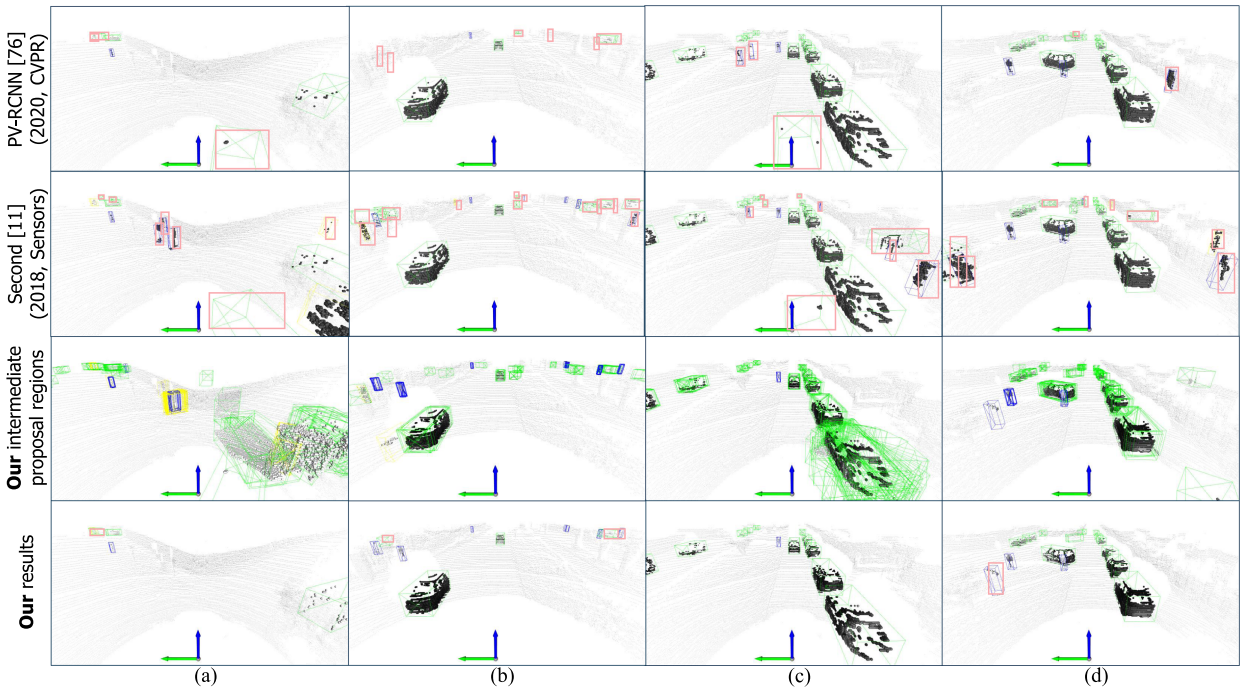



Fig. 8. Qualitative Comparison Between Selected Methods and *EMOS*. We select four representative scenarios: (a) intersection, (b) standard, (c) narrow street, and (d) dense surrounding. The first two rows show the detection results of PV-RCNN [90] and SECOND [53], respectively. The third row illustrates the proposal regions generated by our *AMDB*, while the fourth row presents the final detection results of *EMOS*. Spherical markers highlight voxels within bounding boxes. In the last two rows representing our model, darker spheres indicate voxels with higher confidence scores, while lighter spheres correspond to lower confidence scores. In contrast, the spheres in the first two rows (baselines) are colored solely for visualization purposes. Compared to the two baselines, *EMOS* clearly reduces both false positive and false negative detections.

by lower efficiency utility scores. Across both datasets, *EMOS* attains the highest joint accuracy–efficiency utility scores, reaching $(\mathcal{U}_a = 82.3, \mathcal{U}_e = 59.6)$ on KITTI and $(\mathcal{U}_a = 73.4, \mathcal{U}_e = 178.5)$ on nuScenes. These results demonstrate that the utility-based expert diagnosis provides consistent evaluation across datasets, indicating that the proposed framework can be applied reliably in cross-dataset settings.

*Qualitative Results and Analysis:* We examine four representative driving scenarios from the KITTI validation set: *intersection*, *urban road*, *narrow street*, and *dense traffic*, as shown in Fig. 8. The detection results of *EMOS* are compared against baseline methods including PV-RCNN and SECOND. As shown in Fig. 8, SECOND produces multiple false positives and false negatives across scenarios, such as false positives for pedestrians and cars at intersections, car false positives on urban roads, pedestrian false negatives on narrow streets, and pedestrian false positives in dense traffic. PV-RCNN exhibits a similar error distribution, with frequent false positives across categories, particularly for bicycles and cars, while yielding only a modest reduction in false negatives. Overall, both baselines show higher

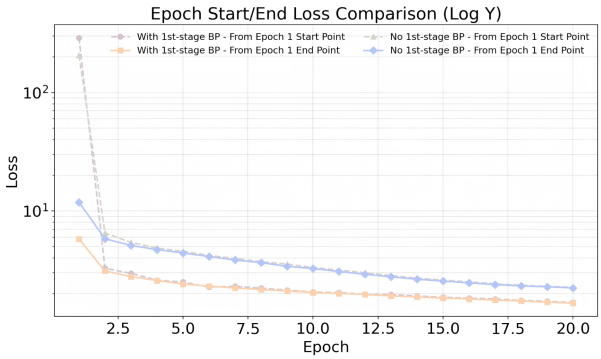
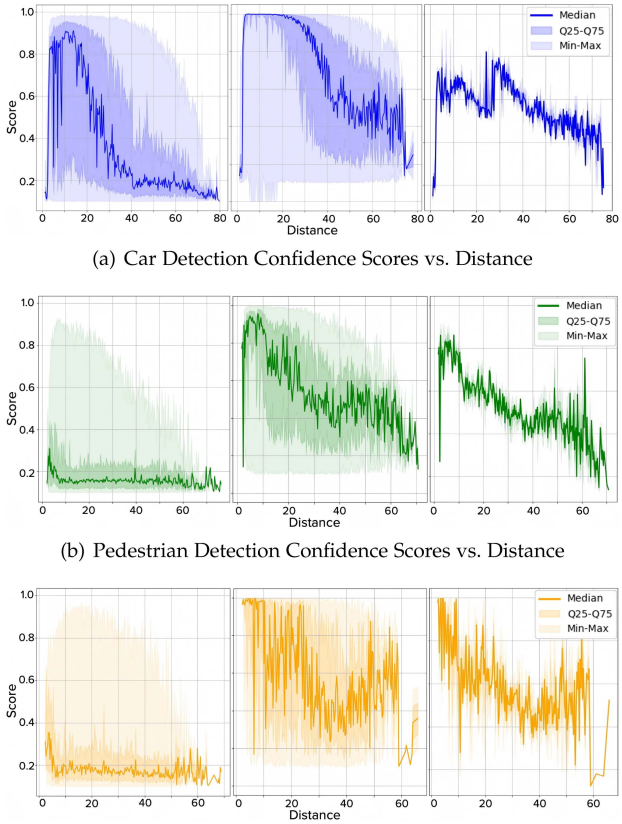



Fig. 9. Ablation Study on Hierarchical Training. We report the training loss over the first 20 epochs, measured at both the *start point* and *end point* of each epoch under two settings: (i) training with first-stage backpropagation and (ii) training without first-stage backpropagation. The "start point" denotes the loss at the beginning of each epoch, while the "end point" denotes the loss at the end of the same epoch. As shown, enabling first-stage backpropagation leads to faster and more stable convergence, whereas disabling first-stage backpropagation results in slower convergence and a higher final loss.

TABLE V
ABLATION STUDY OF DIFFERENT EXPERT ENSEMBLE STRATEGIES. THE TRI-EXPERT COMBINATION (*EPEs* + *LAPEs* + *FAPEs*) ACHIEVES THE BEST IN BOTH ACCURACY UTILITY SCORE AND EFFICIENCY UTILITY SCORE, DEMONSTRATING THAT THE COMPLEMENTARY EXPERTS STRATEGY YIELD SUPERIOR PERFORMANCE.

| E **Ensemble Strategy** | **Model Sizes (MB)** | **mAP** ↑ | **NDS** ↑ |
|---|---|---|---|
| 3 × *EPEs* | 32.00 | 58.0 | 66.7 |
| *EPEs* + *LAPEs* | 112.0 | 67.5 | 71.2 |
| *EPEs* + *FAPEs* | 337.0 | 70.5 | 73.8 |
| *LAPEs* + *FAPEs* | 359.0 | 71.3 | 74.2 |
| *EPEs* + *LAPEs* + *FAPEs* | 333.3 | 71.7 | 74.5 |




(a) Car Detection Confidence Scores vs. Distance

(b) Pedestrian Detection Confidence Scores vs. Distance

(c) Cyclist Detection Confidence Scores vs. Distance

Fig. 10. Detection Confidence Scores v.s. Object Distance. Left to right in each subfigure: (1) *the Efficiency-Prioritized Expert*, (2) *the Lightweight Accuracy-Prioritized Expert*, and (3) the *EMOS* system, on the validation set. Three well-trained experts keep the confidence high with minimal variance across distances.

error rates, whereas *EMOS* demonstrates fewer false positives and false negatives across diverse traffic scenarios.

### D. Ablation Study

*Hierarchical Training:* The training curves of the model with and without the hierarchical training strategy are shown in Fig. 9, illustrating a measurable improvement in convergence speed during the first 20 epochs when hierarchical training is applied. This acceleration can be attributed to the direct supervisory signals provided to the *AMDB*, which enable it to adapt and focus on regions that are likely to contain objects at earlier stages. In contrast, without such supervision, the *AMDB* relies solely on delayed feedback from the detection head, thereby limiting the early identification of informative regions.

*Experts Ensembling:* We analyze ensemble strategies by examining the utilization and per-class performance reported in Table VI. The results show that different classes exhibit distinct feature characteristics, resulting in varying expert utilization. For geometrically simple and frequently observed classes such as *car* (42,821 instances), *EPEs* is selected in 13.6% of cases and achieves accuracy of 88.4 AP and 85.8 NDS. Similarly, for rigid structures like *barrier*, *EPEs* account for 14.9% of selections with mAP 72.7. These results indicate that smaller experts capture regular patterns with limited intra-class variance. In contrast, categories with higher appearance variance or stronger occlusion, such as *bicycle* and *motorcycle*, exhibit higher utilization of larger experts. For instance, *FAPEs* account for 65.6% of selections in the *bicycle* category and achieve 70.1 mAP and 77.5 NDS, compared with 56.8 mAP and 68.8 NDS for *EPEs*. Likewise, for *motorcycle*, *FAPEs* are selected in 67.3% of cases and achieve higher accuracy (85.1 mAP, 82.6 NDS) than smaller experts.

As shown in Table V, pairwise ensembles lead to unbalanced expert utilization. For example, *EPEs+FAPEs* achieves relatively high accuracy (70.5 mAP, 73.8 NDS) but invokes *EPEs* infrequently, reducing efficiency. Similarly, *LAPEs+FAPEs* attains 71.3 mAP and 74.2 NDS, but requires a large combined model size of 359 MB. In comparison, the tri-expert ensemble (*EPEs* + *LAPEs* + *FAPEs*) yields 71.7 mAP and 74.5 NDS with a moderate model size of 333 MB. This configuration utilizes the complementary contributions of all three experts: *EPEs* contribute efficiency, *LAPEs* provide balanced accuracy, and *FAPEs* improve performance in complex scenarios. By routing scenes to the corresponding expert, the tri-expert ensemble achieves a balanced trade-off between efficiency and accuracy across diverse driving conditions.

TABLE VI
THE STATISTICS OF UTILIZATION AND PERFORMANCE OF DIFFERENT EXPERTS DURING EXPERIMENT ON NUSCENES. AS SHOWN IN THE RESULTS, EPES WITH 32 MB SIZE ACHIEVE THE BEST AP (AVERAGE PRECISION) AND NDS IN SOME SCENARIOS AMONG ALL THREE EXPERTS (E.G. TRAILER), WHICH HIGHLIGHTS THE EFFECTIVENESS OF MOE FRAMEWORK IN RESOLVING THE DILEMMA OF ACCURACY AND EFFICIENCY IN 3D OBJECT DETECTION.

| Experts | Classes | Barrier | Bicycle | Bus | Car | Construction | Motorcycle | Pedestrian | Traffic cone | Trailer | Truck |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | Total number | 14481 | 1043 | 1745 | 42821 | 538 | 1310 | 20745 | 7359 | 1448 | 6665 |
| EPEs (LiDAR-only) | Utilization (%) | 14.9 | 12.8 | 13.4 | 13.6 | 13.2 | 10.5 | 12.2 | 10.8 | 14.8 | 12.6 |
| | AP | 72.7 | 56.8 | 81.0 | 88.4 | 31.8 | 71.2 | 87.7 | 77.9 | 53.6 | 66.7 |
| | NDS | 61.0 | 68.8 | 78.8 | 85.8 | 40.7 | 74.1 | 83.6 | 54.8 | 61.9 | 73.6 |
| LAPEs (Multimodal) | Utilization (%) | 29.7 | 21.6 | 21.8 | 28.0 | 25.3 | 22.3 | 26.1 | 22.0 | 38.2 | 29.4 |
| | AP | 78.4 | 66.7 | 80.5 | 88.7 | 29.1 | 80.8 | 89.5 | 78.2 | 50.0 | 66.1 |
| | NDS | 64.0 | 75.1 | 78.7 | 86.1 | 41.8 | 78.1 | 84.2 | 54.9 | 60.7 | 73.2 |
| FAPEs (Multimodal) | Utilization (%) | 55.5 | 65.6 | 64.9 | 58.4 | 61.5 | 67.3 | 61.7 | 67.2 | 47.0 | 58.1 |
| | AP | 77.7 | 70.1 | 80.5 | 90.4 | 36.7 | 85.1 | 90.0 | 81.3 | 44.3 | 71.5 |
| | NDS | 63.8 | 77.5 | 80.3 | 87.6 | 46.4 | 82.6 | 84.8 | 56.3 | 57.0 | 76.2 |

TABLE VII
ABLATION OF EMOS ON NUSCENES. SUMMARY OF UTILIZATION, MAP, AND NDS ACROSS DATA-PROCESSING STRATEGIES. FOR MAP AND NDS, VALUES ARE REPORTED BEFORE AND AFTER OPTIMIZATION: LIGHT-BLUE CELLS SHOW RESULTS AFTER MULTI-STAGE TRAINING TO IMPROVE TAIL-SAMPLE PERFORMANCE (SECTION III-E); PLAIN CELLS SHOW RESULTS BEFORE OPTIMIZATION.

| Data Processing Strategy | Case # | mAP↑ | NDS↑ |
|---|---|---|---|
| AMDB+EPEs (LiDAR-only, 32MB) | 938 | 0.672 | 0.705 |
| | | 0.693 | 0.723 |
| AMDB+LAPEs (159MB) | 1613 | 0.691 | 0.721 |
| | | 0.714 | 0.738 |
| AMDB+FAPEs (496MB) | 3468 | 0.715 | 0.745 |
| | | 0.728 | 0.753 |

*Multiscale Pooling and ONNX-based Optimization:* We analyze the effect of multiscale pooling and ONNX-level optimization on inference latency. On the KITTI dataset, each validation sample contains on average 20,000 LiDAR points. Under standard fusion, the model processes 20,000 LiDAR feature voxels together with approximately 2,000 thousand camera feature voxels, resulting in a two-orders-of-magnitude increase in computation compared to LiDAR alone. The proposed multiscale pooling fusion strategy reduces the camera features to about 60,000 while retaining all 20,000 LiDAR feature voxels, thereby reducing redundant computation. Furthermore, applying ONNX-based system optimization decreases the end-to-end inference time from 6064.2 ms to 397.3 ms, reducing latency to a range suitable for deployment on resource-constrained platforms.

## VI. LIMITATIONS

Our current work primarily focuses on optimizing the algorithmic framework and deployment module. To achieve a more comprehensive solution for algorithm–hardware co-optimization, we will explore topics related to open-vocabulary datasets and hardware utility maximization. We will also investigate integrating language experts into the MoE architecture for open-vocabulary detection, aligning these open-vocabulary experts with existing experts [58], [100], [101]. For hardware utility maximization, we will incorporate SOTA load-balancing techniques [102], [103] to dynamically allocate computational resources across different threads over varying time intervals. We will also explore implementing our strategy on heterogeneous accelerator systems, such as Jetson equipped with both GPU and DLA. This dynamic resource scheduling is designed to optimize hardware utilization, minimize latency, and ensure efficient workload distribution. In particular, we will investigate adaptive resource management strategies that leverage real-time performance monitoring and predictive modeling to allocate computing power more effectively. Such strategies will enable the system to respond dynamically to fluctuations in computational demand, thereby achieving an optimal balance between energy consumption and processing speed.

## VII. CONCLUSION

We presented *EMOS*, a novel multimodal MoE framework that achieves both superior accuracy and efficiency in 3D object detection tasks. By integrating the adaptive MoE routing mechanism, the novel hierarchical training strategy, and system optimization including multiscale pooling and computational graph optimization, *EMOS* dynamically adapts to diverse traffic scenarios, ensuring efficiency without compromising accuracy. *EMOS* demonstrates SOTA accuracy and efficiency across both KITTI and nuScenes benchmarks. On KITTI, it improves average recognition accuracy by 3.17% while running 2.6 times faster on Jetson. On nuScenes, it surpasses accuracy-oriented SOTA models with gains of 0.2% mAP and 0.5% NDS, using only 66% of the model size and achieving a 15.35 times speedup on Jetson. These results confirm that EMOS effectively balances accuracy and efficiency, making it a practical solution for real-time autonomous driving on resource-constrained edge platforms. Future work includes designing a more precise and adaptive MoE switch to further enhance expert selection across varying scenarios, as well as exploring additional deployment optimizations to further minimize latency while preserving accuracy, thereby reinforcing the practicality of *EMOS* for real-time autonomous driving systems.

## REFERENCES

[1] H. Mao, X. Yang, and B. Dally, " A delay metric for video object detection: What average precision fails to tell," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, Los Alamitos, CA, USA: IEEE Computer Society, 2019, pp. 573–582, doi: 10.1109/ICCV.2019.00066.

[2] H. Caesar et al., "nuscenes: A multimodal dataset for autonomous driving," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 11621–11 631.

[3] A. Kendall, Y. Gal, and R. Cipolla, "Multi-task learning using uncertainty to weigh losses for scene geometry and semantics," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 7482–7491.

[4] A. Esteva et al., "A guide to deep learning in healthcare," *Nature Med.*, vol. 25, no. 1, pp. 24–29, 2019.

[5] A. G. Howard et al., "Mobilenets: Efficient convolutional neural networks for mobile vision applications," 2017, *arXiv: 1704.04861*.

[6] M. Tan and Q. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 6105–6114.

[7] J. Bai et al., "ONNX: Open neural network exchange," 2019. [Online]. Available: https://onnx.ai

[8] S. Han et al., "EIE: Efficient inference engine on compressed deep neural network," *ACM SIGARCH Comput. Archit. News*, vol. 44, no. 3, pp. 243–254, 2016.

[9] Y. Guo, A. Yao, and Y. Chen, "Dynamic network surgery for efficient DNNs," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 1387–1395.

[10] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? The kitti vision benchmark suite," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2012, pp. 3354–3361.

[11] E. Arnold, O. Y. Al-Jarrah, M. Dianati, S. Fallah, D. Oxtoby, and A. Mouzakitis, "A survey on 3D object detection methods for autonomous driving applications," *IEEE Trans. Intell. Transp. Syst.*, vol. 20, no. 10, pp. 3782–3795, Oct. 2019.

[12] Y. Li et al., "Deep learning for LiDAR point clouds in autonomous driving: A review," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 32, no. 8, pp. 3412–3432, Aug. 2021.

[13] D. Lepikhin et al., "GShard: Scaling giant models with conditional computation and automatic sharding," 2020, *arXiv: 2006.16668*.

[14] W. Cai, J. Jiang, F. Wang, J. Tang, S. Kim, and J. Huang, "A survey on mixture of experts," 2024, *arXiv:2407.06204*.

[15] K. Huang, B. Shi, X. Li, X. Li, S. Huang, and Y. Li, "Multi-modal sensor fusion for auto driving perception: A survey," 2022, *arXiv:2202.02703*.

[16] D. Lepikhin et al., "GShard: Scaling giant models with conditional computation and automatic sharding," 2020, *arXiv: 2006.16668*.

[17] N. Du et al., "Glam: Efficient scaling of language models with mixture-of-experts," in *Proc. Int. Conf. Mach. Learn.*, 2022, pp. 5547–5569.

[18] Y. Zhou et al., "Mixture-of-experts with expertchoice routing," in *Proc. Adv. Neural Inf. Process. Syst.*, S.Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A.Oh, Eds., vol. 35. Curran Associates, Inc., 2022, pp. 7103–7114. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2022/file/2f00ecd787b432c1d36f3de9800728eb-Paper-Conference.pdf

[19] B. Mustafa, C. Riquelme, J. Puigcerver, R. Jenatton, and N. Houlsby, "Multimodal contrastive learning with limoe: The language-image mixture of experts," in *Proc. Adv. Neural Inf. Process. Syst.*, S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, Eds., vol. 35. Curran Associates, Inc., 2022, pp. 9564–9576. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2022/file/3e67e84abf900bb2c7cbd5759bfce62d-Paper-Conference.pdf

[20] G. ShenR. ChenW. ShaoGuan, and L. Nie, "Mome: Mixture of multimodal experts for generalist multimodal large language models," in *Proc. Adv. Neural Inf. Process. Syst.*, A.L. Globerson-Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang, Eds., vol. 37. Curran Associates, Inc., 2024, pp. 42048–42070. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2024/file/4a3a14b9536806a0522930007c5512f7-Paper-Conference.pdf

[21] B. Lin et al., "MoE-LLaVA: Mixture of experts for large vision-language models," 2024, *arXiv:2401.15947*.

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

[23] S. Shi, X. Wang, and H. Li, "PointrCNN: 3D object proposal generation and detection from point cloud," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 770–779.

[24] Y. Chen, S. Liu, X. Shen, and J. Jia, "Fast point R-CNN," in *Proc. 2019 IEEE/CVF Int. Conf. Comput. Vis.*, 2019, pp. 9774–9783. [Online]. Available: https://api.semanticscholar.org/CorpusID:199501855

[25] 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, Aug. 2021.

[26] J. Deng, S. Shi, P. Li, W. Zhou, Y. Zhang, and H. Li, "Voxel R-CNN: Towards high performance voxel-based 3D Object Detection," 2020, *arXiv: 2012.15712*.

[27] Y. Li, Y. Chen, X. Qi, Z. Li, J. Sun, and J. Jia, "Unifying voxel-based representation with transformer for 3D object detection," in *Adv. Neural Inf. Process. Syst.*, 2022, pp. 18442–18455.

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

[29] Z. Liu et al., "Bevfusion: Multi-task multi-sensor fusion with unified bird's-eye view representation," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2023, pp. 2774–2781.

[30] NVIDIA Corporation, "NVIDIA TensorRT developer guide," 2024, Accessed: 2025–Feb.–13. [Online]. Available: https://docs.nvidia.com/deeplearning/tensorrt/latest/index.html

[31] A. Sabne, "XLA: Compiling machine learning for peak performance," 2020.

[32] T. Chen et al., "{TVM }: An automated { End-to-End} optimizing compiler for deep learning," in *Proc. 13th USENIX Symp. Operating Syst. Des. Implementation*, 2018, pp. 578–594.

[33] Z. Song et al., "Robustness-aware 3D object detection in autonomous driving: A review and outlook," *IEEE Trans. Intell. Transp. Syst.*, vol. 25, no. 11, pp. 15407–15436, Nov. 2024.

[34] F. Chabot, M. Chaouch, J. Rabarisoa, C. Teulière, and T. Chateau, "Deep manta: A coarse-to-fine many-task network for joint 2D and 3D vehicle analysis from monocular image," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 1827–1836.

[35] P. Li, H. Zhao, P. Liu, and F. Cao, "RTM3D: Real-time monocular 3D detection from object keypoints for autonomous driving," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2020, pp. 644–660.

[36] Z. Min, B. Zhuang, S. Schulter, B. Liu, E. Dunn, and M. Chandraker, "Neurocs: Neural nocs supervision for monocular 3D object localization," in *Proc. 2023 IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2023, pp. 21404–21414.

[37] R. Zhang et al., "MonoDETR: Depth-guided transformer for monocular 3D object detection," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2022, pp. 9121–9132.

[38] G. Brazil, A. Kumar, J. Straub, N. Ravi, J. Johnson, and G. Gkioxari, "Omni3D: A large benchmark and model for 3D object detection in the wild," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Vancouver, Canada: IEEE, Jun. 2023, pp. 13154–13164.

[39] L. Peng, X. Wu, Z. Yang, H. Liu, and D. Cai, "DID-M3D: Decoupling instance depth for monocular 3D object detection," in *Proc. Eur. Conf. Comput. Vis.*, 2022, pp. 71–88.

[40] W. Zhang, D. Liu, C. Ma, and W. Cai, "Alleviating foreground sparsity for semi-supervised monocular 3D object detection," in *Proc. 2024 IEEE/CVF Winter Conf. Appl. Comput. Vis.*, 2024, pp. 7527–7537.

[41] Z. Wu, Y. Gan, Y. Wu, R. Wang, X. Wang, and J. Pu, "FD3D: Exploiting foreground depth map for feature-supervised monocular 3D object detection," in *Proc. AAAI Conf. Artif. Intell.*, 2024, pp. 6189–6197.

[42] L. Wang et al., "Depth-conditioned dynamic message propagation for monocular 3D object detection," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 454–463.

[43] H. F. Yang, J. Cai, C. Liu, R. Ke, and Y. Wang, "Cooperative multi-camera vehicle tracking and traffic surveillance with edge artificial intelligence and representation learning," *Transp. Res. Part C: Emerg. Technol.*, vol. 148, 2023, Art. no. 103982.

[44] Z. Shen, X. Song, Y. Dai, D. Zhou, Z. Rao, and L. Zhang, "Digging into uncertainty-based pseudo-label for robust stereo matching," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, pp. 14301–14320, Dec. 2023.

[45] G. Xu, X. Wang, X. Ding, and X. Yang, "Iterative geometry encoding volume for stereo matching," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2023, pp. 21919–21 928.

[46] G. Xu, Y. Wang, J. Cheng, J. Tang, and X. Yang, "Accurate and efficient stereo matching via attention concatenation volume," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 46, no. 4, pp. 2461–2474, Apr. 2024.

[47] X. Cheng et al., "Hierarchical neural architecture search for deep stereo matching," in *Proc. Adv. Neural Inf. Process. Syst.*, 2020, pp. 22158–22169.

[48] H. Yang, J. Cai, M. Zhu, C. Liu, and Y. Wang, "Traffic-informed multi-camera sensing (TIMS) system based on vehicle re-identification," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 10, pp. 17189–17200, 2022.

[49] Q. Chen, L. Sun, E. Cheung, and A. Yuille, "Every view counts: Cross-view consistency in 3D object detection with hybrid-cylindrical-spherical voxelization," in *Proc. 34th Int. Conf. Neural Inf. Process. Syst.*, Red Hook, NY, USA: Curran Associates Inc., 2020, pp. 21224–21235.

[50] 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," 2019, *arXiv: 1907.03670*.

[51] 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. 2019 IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 12689–12 697.

[52] B. Li, "3D fully convolutional network for vehicle detection in point cloud," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2017, pp. 1513–1518.

[53] Y. Yan, Y. Mao, and B. Li, "SECOND: Sparsely embedded convolutional detection," *Sensors*, vol. 18, no. 10, 2018, Art. no. 3337. [Online]. Available: https://www.mdpi.com/1424-8220/18/10/3337

[54] S. Shi et al., "PV-RCNN: Point-voxel feature set abstraction for 3D object detection," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 10526–10535.

[55] 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.*, 2020, pp. 11037–110 45.

[56] Y. Wang et al., "Pillar-based object detection for autonomous driving," in *Proc. 16th Eur. Conf. Comput. Vis.*, Glasgow, U.K., 2020, pp. 18–34, doi: 10.1007/978-3-030-58542-6_2.

[57] T. Yin, X. Zhou, and P. Krähenbühl, "Center-based 3D object detection and tracking," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2021, pp. 11779–11788.

[58] Z. HuangX. LiuChen, and X. Bai, "EPNet: Enhancing point features with image semantics for 3D object detection," in *Proc. Eur. Conf. Comput. Vis.*, A.H. VedaldiT. BischofBrox, and J.-M. Frahm, Eds. Cham: Springer, 2020, pp. 35–52.

[59] S. Vora, A. H. Lang, B. Helou, and O. Beijbom, "Pointpainting: Sequential fusion for 3D object detection," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 4603–4611.

[60] R. Nabati and H. Qi, "Centerfusion: Center-based radar and camera fusion for 3D object detection," in *Proc. IEEE Winter Conf. Appl. Comput. Vis.*, 2021, pp. 1526–1535.

[61] C. Feng, C. Xiang, X. Xie, Y. Zhang, M. Yang, and X. Li, "HPV-RCNN: Hybrid point–voxel two-stage network for LiDAR-based 3-D object detection," *IEEE Trans. Computat. Social Syst.*, vol. 10, no. 6, pp. 3066–3076, Dec. 2023.

[62] Z. Wang, W. Zhan, and M. Tomizuka, "Fusing bird's eye view LiDAR point cloud and front view camera image for 3D object detection," in *Proc. IEEE Intell. Veh. Symp.*, 2018, pp. 1–6.

[63] D. Xu, D. Anguelov, and A. Jain, "Pointfusion: Deep sensor fusion for 3D bounding box estimation," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2018, pp. 244–253.

[64] 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.*, 2023, pp. 13488–13 498.

[65] Y. Li et al., "Deepfusion: LiDAR-camera deep fusion for multi-modal 3D object detection," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2022, pp. 17161–17170.

[66] C. Zhang, H. Wang, L. Chen, Y. Li, and Y. Cai, "Mixedfusion: An efficient multimodal data fusion framework for 3-D object detection and tracking," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 36, no. 1, pp. 1842–1856, Jan. 2025.

[67] Z. Yang, N. Song, W. Li, X. Zhu, L. Zhang, and P. H. S, "Deepinteraction : Multi-modality interaction for autonomous driving," 2024, *arXiv:2408.05075*.

[68] X. Wu et al., "Sparse fuse dense: Towards high quality 3D detection with depth completion," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2022, pp. 5408–5417.

[69] J. Yan et al., "Cross modal transformer: Towards fast and robust 3D object detection," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2023, pp. 18222–18232.

[70] H. Wu, C. Wen, S. Shi, X. Li, and C. Wang, "Virtual sparse convolution for multimodal 3D object detection," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2023, pp. 21653–21 662.

[71] G. Xie, Z. Chen, M. Gao, M. Hu, and X. Qin, "PPF-det: Point-pixel fusion for multi-modal 3D object detection," *IEEE Trans. Intell. Transp. Syst.*, vol. 25, no. 6, pp. 5598–5611, Jun. 2024.

[72] Z. Song et al., "RoboFusion: Towards robust multi-modal 3D obiect detection via SAM," 2024, *arXiv:2401.03907*.

[73] Z. Song, C. Jia, L. Yang, H. Wei, and L. Liu, "GraphAlign : An accurate feature alignment by graph matching for multi-modal 3D object detection," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 34, no. 4, pp. 2619–2632, Apr. 2024.

[74] O. Community, "Onnx: Open neural network exchange," 2024. [Online]. Available: https://github.com/onnx/onnx

[75] C. Lattner et al., "MLIR: A compiler infrastructure for the end of moore's law," 2020, *arXiv: 2002.11054*.

[76] C. Lattner and V. Adve, "LLVM: A compilation framework for lifelong program analysis & transformation," in *Proc. Int. Symp. Code Gener. Optim.*, 2004, pp. 75–86.

[77] V. Zunin, "Intel openvino toolkit for computer vision: Object detection and semantic segmentation," in *Proc. Int. Russian Automat. Conf.*, 2021, pp. 847–851.

[78] J. Li et al., "oneDNN graph compiler: A hybrid approach for high-performance deep learning compilation," in *Proc. IEEE/ACM Int. Symp. Code Gener. Optim.*, 2024, pp. 460–470.

[79] D. P. Kingma and P. Dhariwal, "Glow: Generative flow with invertible 1x1 convolutions," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 10215–10224.

[80] J. Ansel et al., "Pytorch 2: Faster machine learning through dynamic python bytecode transformation and graph compilation," in *Proc. 29th ACM Int. Conf. Architectural Support Program. Lang. Operating Syst.*, 2024, pp. 929–947.

[81] J. Huang, G. Huang, Z. Zhu, Y. Ye, and D. Du, "BEVDet: High-performance multi-camera 3D object detection in bird-eye-view," 2021, *arXiv:2112.11790*.

[82] R. A. Jacobs, M. I. Jordan, S. J. Nowlan, and G. E. Hinton, "Adaptive mixtures of local experts," *Neural Computation*, vol. 3, no. 1, pp. 79–87, 1991.

[83] J. Cai, Y. Wang, and J.-N. Hwang, "ACE: Ally complementary experts for solving long-tailed recognition in one-shot," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2021, pp. 112–121.

[84] B. Graham, "Sparse 3D convolutional neural networks," 2015, *arXiv:1505.02890*.

[85] H. A. Jamrozik et al., "Reducing network latency using subpages in a global memory environment," *ACM SIGOPS Operating Syst. Rev.*, vol. 30, no. 5, pp. 258–267, 1996.

[86] J. Hromkovič, *Communication Complexity and Parallel Computing*. Berlin, Germany: Springer, 2013.

[87] H. Qin, Q. Li, J. Speiser, P. Kraft, and J. Ousterhout, "Arachne:{Core-Aware} thread management," in *Proc. 13th USENIX Symp. Operating Syst. Des. Implementation*, 2018, pp. 145–160.

[88] C. Deng, Y. Sui, S. Liao, X. Qian, and B. Yuan, "GoSPA: An energy-efficient high-performance globally optimized sparse convolutional neural network accelerator," in *Proc. ACM/IEEE 48th Annu. Int. Symp. Comput. Archit.*, 2021, pp. 1110–1123.

[89] T. Guan et al., "M3DeTR: Multi-representation, multi-scale, mutual-relation 3D object detection with transformers," in *Proc. IEEE/CVF Winter Conf. Appl. Comput. Vis.*, 2022, pp. 772–782.

[90] S. Shi et al., "PV-RCNN: Point-voxel feature set abstraction for 3D object detection," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 10529–10538.

[91] J. Liao, Y. Liu, Y. Piao, J. Su, G. Cai, and Y. Wu, "GLE-net: A global and local ensemble network for aerial object detection," *Int. J. Comput. Intell. Syst.*, vol. 15, no. 1, 2022, Art. no. 2.

[92] H. Sheng et al., "Improving 3D object detection with channel-wise transformer," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2021, pp. 2743–2752.

[93] H. Wu, C. Wen, W. Li, X. Li, R. Yang, and C. Wang, "Transformation-equivariant 3D object detection for autonomous driving," in *Proc. AAAI Conf. Artif. Intell.*, 2023, pp. 2795–2802.

[94] X. Li et al., "LoGoNet: Towards accurate 3D object detection with local-to-global cross-modal fusion," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2023, pp. 17524–17534.

[95] Y. Li et al., "Voxel field fusion for 3D object detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2022, pp. 1110–1119.

[96] 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.*, 2023, pp. 21674–21 683.

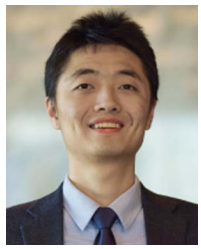
[97] X. Lin, Z. Pei, T. Lin, L. Huang, and Z. Su, "Sparse4D v3: Advancing end-to-end 3D detection and tracking," 2023, *arXiv:2311.11722*.
[98] Z. Lin, Z. Liu, Y. Wang, L. Zhang, and C. Zhu, "RCBEVDet: Toward high-accuracy radar-camera fusion 3D perception network," 2024, *arXiv:2409.04979*.
[99] X. Bai et al., "Transfusion: Robust LiDAR-camera fusion for 3D object detection with transformers," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2022, pp. 1090–1099.
[100] Y. Zhao, J. Lin, and R. W. Lau, "Hierarchical cross-modal alignment for open-vocabulary 3D object detection," in *Proc. AAAI Conf. Artif. Intell.*, 2025, pp. 10501–10509.
[101] T. Yang, Y. Ju, and L. Yi, "ImOV3D: Learning open-vocabulary point clouds 3D object detection from only 2D images," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2024, Art. no. 7.
[102] Y. Dai, D. Xu, S. Maharjan, and Y. Zhang, "Joint load balancing and offloading in vehicular edge computing and networks," *IEEE Internet Things J.*, vol. 6, no. 3, pp. 4377–4387, Jun. 2019.
[103] J. Zhang, H. Guo, J. Liu, and Y. Zhang, "Task offloading in vehicular edge computing networks: A load-balancing solution," *IEEE Trans. Veh. Technol.*, vol. 69, no. 2, pp. 2092–2104, Feb. 2020.

**Linshen Liu** received the MS degree in electrical and computer engineering from the University of Texas at Austin, Austin, TX, USA. He is currently working toward the PhD degree with the Department of Civil and Systems Engineering, Johns Hopkins University, Baltimore, MD, USA, advised by Prof. Hao Frank Yang. His research interests include efficient machine learning systems and edge AI for autonomous driving, with an emphasis on real-time, resource-constrained perception, and decision making.

**Pu Wang** received the MS degree in computer science from New York University, in 2025. He is currently working toward the PhD degree with the Department of Civil and Systems Engineering, Johns Hopkins University, Baltimore, MD, USA, advised by Prof. Hao Frank Yang. His research interests include data-centric AI and the applications of generative AI to complex systems, such as autonomous driving. He also investigates approaches for aligning diverse data modalities within multimodal large language models toward more coherent and effective multimodal AI systems.

**Guanlin Wu** received the BE degree in computer science and engineering from the Chinese University of Hong Kong, Shenzhen, China, and the MS degree in systems engineering from Johns Hopkins University, Baltimore, MD, USA. His research interests include computer vision and multimodal large language models.

**Junyue Jiang** received dual bachelor's degrees in electrical and computer engineering from Zhejiang University and the University of Illinois Urbana-Champaign. He is currently working toward the PhD degree with the Department of Civil and Systems Engineering, Johns Hopkins University, Baltimore, MD, USA, advised by Prof. Hao Frank Yang. His research interests include sustainable and efficient computing for machine learning systems, with a focus on energy-efficient decision making for edge AI and hardware–software co-optimization in smart transportation.



**Hao Frank Yang** received the PhD degree in civil engineering (transportation) from the University of Washington. He is an assistant professor with the Department of Civil and Systems Engineering and a member of the Data Science and AI Institute, and Institute of Assured Autonomy at Johns Hopkins University. His research focuses on decision-focused computing methods and systems, particularly for real-world applications in transportation and public health. On the algorithmic side, his work spans foundation model adaptation with multimodal representation learning, symbolic reasoning, and data optimization. On the systems side, he works on MoE–based architectures innovations, including multimodal, safety-guaranteed, and neuromorphic MoE designs, to achieve efficient and robust performance. He was a research scientist with the NSF AI Institute for Edge Computing and the Department of Electrical and Computer Engineering at Duke University. He received dual bachelor's degrees in electrical and computer (telecommunication) engineering from the Beijing University of Posts and Telecommunications and the University of London. He has published more than 45 journal papers and conference proceedings in Nature Computational Science, Nature Communications, CVPR, ICLR, NeurIPS, ICCV, Information Fusion, Transportation Research Part C, and *IEEE Transactions on Intelligent Transportation Systems*, among others.