Multimodal Mixture-of-Experts Computing System on Edge for Efficient and Robust 3D Detection in Autonomous Driving

Linshen Liu[1,+], Boyan Su[2,+], Guanlin Wu[1], Junyue Jiang[1], Cong Guo[3], and Hao Frank Yang[1,*]

[1]Department of Civil and System Engineering, Johns Hopkins University
[2]Department of Applied Mathematics and Statistics, Johns Hopkins University
[3]Department of Electrical and Computer Engineering, Duke University
[*]Corresponding author, Email: haofrankyang@jhu.edu
[+]these authors contributed equally to this work

ABSTRACT

Autonomous vehicles (AVs) rely on accurate and efficient perception modules, responsible for critical tasks of object detection and tracking in real time. While high-precision models ensure reliability, the computational overhead can hinder real-time performance, leading to safety concerns. This paper introduces EMC2, an Edge-based Mixture-of-Experts Collaborative Computing architecture that addresses the challenge of balancing accuracy and latency in autonomous driving perception. Our approach leverages multimodal sensor data and scenario-aware routing, activating specialized sub-models only when needed, thus achieving both high accuracy and low latency across diverse driving conditions. Specifically, EMC2 comprises an Adaptive Multimodal Data Bridge for efficient preprocessing, a Scenario-aware MoE Switch that allocates resources based on latency and accuracy demands, and four Complementary Experts optimized for different scenarios. Our design includes a Hierarchical Back-Propagation and a Multiscale Pooling layer, effectively adapting model complexity to traffic conditions. Experimental results demonstrate state-of-the-art performance in mutimodal 3D object detection task, especially in difficult scenarios with dense traffic, poor lighting, and frequent unpredictable events, achieving up to 1.42%, 3.28%, and 6.03% higher accuracy for cars, pedestrians, and bicycles, respectively, compared to current best models. Furthermore, EMC2 delivers superior efficiency, processing inputs 159.06% faster on Jetson platforms than baseline methods while maintaining performance. By leveraging both comprehensive multimodal information and intelligent expert selection, EMC2 offers an effective, edge-adaptive, and real-time perception solution for autonomous driving systems, paving the way for safer and more flexible perception systems, thereby ensuring robust performance and timely reactions in real-world driving environments.

1 Introduction

Traffic safety is the top priority for both human drivers and Autonomous Driving Systems (ADS). In ADS, the perception module serves as the 'eyes' of an autonomous vehicle (AV), gathering information about the surrounding environment and converting it into critical instructions for downstream units such as decision-making and control. The conceptual safety of an AV perception system can be delineated into two fundamental criteria: accuracy and efficiency. Accuracy ensures reliable object detection and tracking results, providing a trustworthy understanding of the environment. Efficiency, i.e., low perception latency, is equally critical, as real-time decision-making and control rely on rapid processing. Perception latency, the sum of computational and algorithmic delays [1], directly impacts an ADS's ability to react to dynamic environments. Studies suggest that an ADS must process sensor data and execute control actions within 100 milliseconds to ensure timely and safe maneuvering [2]. **Accuracy and latency together define a critical safety threshold: an accurate but slow system fails to react in time, while a fast but inaccurate system makes unsafe decisions.**

However, achieving both high accuracy and low latency remains a fundamental challenge. As illustrated in Fig. 1, existing approaches often lean toward one side: some prioritize accuracy at the expense of inference speed, while others optimize

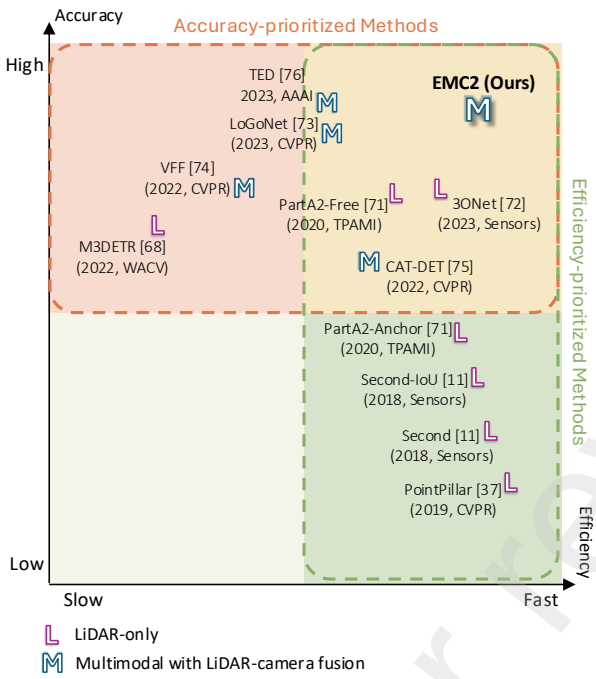


Figure 1. Performance of 3D object detection models in terms of accuracy and inference efficiency on the KITTI dataset. The results indicate that most methods improve accuracy at the expense of inference efficiency. The proposed EMC2 achieves state-of-the-art performance by employing different experts in a scenario-aware manner.

efficiency but compromise detection performance. This trade-off stems from multiple factors. First, computational resources are inherently limited, making it difficult to deploy high-accuracy models with increasingly sophisticated architectures in real-time settings. Second, conventional perception architectures lack adaptive mechanisms to allocate resources dynamically based on situations, resulting in excessive computation in simple scenarios or insufficient capacity in challenging conditions. Beyond computational constraints, diverse real-world driving scenarios further complicate the challenge. Urban environments present dense, dynamic surroundings with frequent interactions among vulnerable road users, requiring rapid and precise perception to handle unpredictable movements. Highways, in contrast, feature high-speed traffic in a structured setting, where the system must prioritize long-range detection while ensuring timely responses to sudden lane changes or braking events. Even within the same scene, recognition priorities vary: distant objects do not need frame-by-frame precision but must be accurately detected upon reaching the relevant decision-making range. These factors make it difficult for a single unimodal perception model to achieve both high accuracy and low latency, particularly given their intrinsic limitations aforementioned.

In academic terms, an ideal approach delivers results within a reasonable accuracy bound while minimizing processing time. A common solution is a single high-precision algorithm with multimodal data, typically using large-scale CNN-based models that take LiDAR and camera as inputs [3]. This multimodal approach significantly enhances perception accuracy by leveraging the complementary strengths of different sensors: LiDAR provides reliable 3D spatial information, unaffected by lighting conditions, but lacks texture details; cameras offer rich visual information but struggle with depth estimation and low-light conditions. However, despite these accuracy gains, this one-size-fits-all architecture is still computationally inefficient and lacks adaptability to different efficiency demands, especially on edge platforms [4]. Even with machine learning compilation frameworks that optimize hardware utilization [5], such systems still perform unnecessary computations in scenarios that do not require intensive processing, wasting both time and resources. For instance, in straightforward conditions with clear visibility, unexpected objects or movements can be identified with minimal processing, allowing for immediate vehicle response. Yet, the same multimodal system, despite being capable of recognizing these events, would still execute redundant computations, introducing unnecessary latency and delaying critical safety decisions.

That being said, multimodal strategy offers a natural foundation for addressing ADS challenges. Instead of relying on a monolithic model uniformly optimized for all data types and traffic scenarios, Mixture-of-Experts (MoE) [6] selectively activates specialized sub-models, or experts, based on input characteristics. MoE has already demonstrated remarkable success in large-scale applications, particularly in Large Language Models (LLMs), where it routes different tokens to relevant experts, reducing computational redundancy while maintaining high performance [7]. As models scale, MoE enables LLMs to expand capacity while keeping inference costs manageable. However, deploying MoE in autonomous driving (ADS) presents unique challenges. First, unlike LLMs, ADS systems operate under strict real-time constraints, where delays in processing directly impact safety-critical decision-making. Second, while LLMs primarily process single-modality text data, ADS relies on multimodal sensor inputs (LiDAR, camera, radar, etc.), each carrying distinct advantages and computational demands. Third, LLMs can rely on human agents to refine or interpret inputs when necessary, whereas ADS must process raw sensor data autonomously, requiring an end-to-end solution. Given this situation, preprocessing data, prioritizing certain modalities based on scene characteristics, and linking them to the appropriate expert with an optimal routing strategy are essential requirements for deploying MoE architecture in end-to-end ADS.

By leveraging the rich information from multimodal data and context-sensitive MoE routing, we ensure that complex, multimodal models are reserved for intricate conditions, while simpler, faster models relying on single-modality data handle less demanding cases. This approach maximizes accuracy while minimizing latency across diverse scenarios, presenting a transformative opportunity to address the challenges in ADS. To this end, we propose an **E**dge-based **M**oE **C**ollaborative **C**omputing (EMC2) framework – a system-level solution that not only enhances accuracy through multimodal information but also prevents significant latency. Specifically, our contributions are summarized as follows:

- **A Novel Multimodal 3D Object Detection Framework with MoE.** This work introduces EMC2, an MoE-based 3D object detection system that balances **accuracy and efficiency** for LiDAR and camera inputs. By integrating real-world traffic safety requirements with computational constraints, EMC2 optimally selects experts to ensure robust perception with minimal latency, adapting dynamically to different traffic scenarios.
- **Enhanced Accuracy and Robustness via MoE and Hierarchical Training. EMC2** integrates diverse experts: *latency-prioritized*, *LiDAR-only*, and *multimodal accuracy-prioritized* 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. Additionally, a **distribution-aware resampling procedure** improves robustness against real-world data imbalance. 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. EMC2 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 significantly improving inference speed.
- **SOTA Detection Performance, Especially in Hard Scenarios. EMC2** outperforms existing LiDAR-only and multi-modal models, particularly in challenging conditions such as dense traffic, low lighting, and occlusions. Compared to the best current models, EMC2 achieves **+1.42%** (cars), **+3.28%** (pedestrians), and **+6.03%** (bicycles) in hard-level recognition accuracy.
- **SOTA Efficiency and Edge Deployability.** Designed for real-time autonomous driving applications, EMC2 operates **159.06%** faster on Jetson devices, meeting the stringent latency constraints of ADS without sacrificing accuracy. Its efficient expert selection and system-level optimizations enable direct deployment on edge devices, making it practical for resource-limited environments.

2 Related Work

Improving both accuracy and latency for a network system is a highly complex task that requires balancing optimizations across multiple domains. Achieving overall performance gains demands not only in-depth algorithmic improvements but also computer system-level optimizations. We systematically compare current SOTA research at both levels, with detailed results presented in Tab. 1, offering an intuitive understanding of different optimization strategies. The remainder of this chapter provides an in-depth discussion of related work on Object Detection Algorithms, Machine Learning Compilation Platforms, and Mixture of Experts.

2.1 Accuracy-prioritized 3D Object Detection

Camera-only Method. In ADS, monocular and stereo cameras are widely used imaging sensors. For monocular cameras, research divides into three main categories [8]: (1) Prior-guided methods [9], [10], [11], [12], (2) Camera-only approaches [13], [14], [15], and (3) Depth-assisted techniques [16], [17], [18], [19]. For stereo cameras, research falls into two categories: (1) 2D-detection-based methods [20], [21], [22], [23] and (2) Volume-based approaches [24], [25], [26], [27]. 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, resulting in lower performance in 3D object detection compared to LiDAR. Processing dense image signals also demands significant computational resources and time. Consequently, camera-based methods generally underperform compared to LiDAR-only methods in 3D object detection.

LiDAR-only Method. Due to LiDAR's active signal generation attributes with 3D information, it has been widely used in ADS for 3D object detection [28], [29], [30], [31], [32], [33], [34], [35]. Some approaches predict 3D bounding boxes from point clouds by first projecting them onto 3D voxels [36], [37] or pillars [32], while others directly use raw point clouds as input to their models [38], [39], [40]. Anchor boxes are the primary 3D detection heads employed [32], [37], although center-based representations for 3D objects are also adopted [41], [42]. For LiDAR-only methods, accuracy on KITTI car detection has significantly improved: 3DSSD [39], PartA2 [31], and Voxel R-CNN [3] all achieves 85% plus accuracy on easy case, while only 72% higher in hard scenarios, which leading to serious robustness concerns. However, to prevent potential harm from laser emissions to humans, LiDAR sensors are subject to strict limitations on signal intensity. This constraint results in weaker signal reflections when interacting with surfaces at small angles of incidence or with soft materials such as skin and cloth. Consequently, LiDAR methods perform poorly in detecting the "person" class on the KITTI dataset.

Multimodal Fusion Method. Although LiDAR provides accurate and robust point cloud information, its lack of RGB content poses a significant constraint for detecting small objects or items like traffic signs. Consequently, the fusion of LiDAR and camera data has become an increasingly popular research direction in ADS. –To align LiDAR and image features from different dimensions and coordinate systems, two primary approaches have emerged [8]: projection-based and model-based feature alignment. Projection-based methods either project image features onto raw point clouds to preserve the original point cloud representation [43], [44], [45], [46] or focus on fusing features extracted from point clouds and images during the feature extraction stage [47], [48], [49], [50]. Model-based methods, on the other hand, either utilize query-learning to achieve feature alignment through cross-attention mechanisms before fusing features [51], [52], [53], [54] or employ a unified feature space for fusion [55], [56], [57], [58]. Among these multimodal methods, the accuracy on KITTI car detection benchmarks is noteworthy: PPF-Det achieves scores of 89.51, 84.46, and 78.91; RoboFusion reaches 91.75, 84.08, and 80.71; and GraphAlign++ attains 90.98, 83.76, and 80.16 [59], [60], [61].

2.2 Efficiency-prioritized 3D Object Detection

To ensure that the proposed ADS can be deployed in real-world applications in real-time, efficiency-prioritized methods should have a latency lower than 100ms on edge computing devices [2]. However, since most proposed methods have not been tested on edge computing devices such as Jetson, we illustrate methods with latency less than 100ms and specify the GPUs used in this section. Similar to the previous discussion, the performance of camera-only methods is not ideal. The monocular camera-based

Table 1. Comparison of the Characteristics of Existing Efficient ML Systems with Those of EMC2.

Method	Algorithm			System		
	Multimodal Input	Parameter Minimization	Pooling Encoder	CPU Scheduling	Memory Management	Computational Graphic Optimization
VoxelNet [37]	✗	✗	✓	✗	✗	✗
PointRCNN [33]	✗	✗	✓	✗	✗	✗
Fast-Point RCNN [35]	✗	✗	✓	✗	✗	✗
Part-A2 [62]	✓	✗	✓	✗	✗	✗
Voxel R-CNN [3]	✗	✗	✓	✗	✗	✗
Voxel Transformer [55]	✗	✗	✓	✗	✗	✗
Focals Conv [49]	✗	✗	✓	✗	✗	✗
BEV-Fusion [52]	✓	✗	✓	✗	✗	✗
TensorRT [63]	✗	✗	✗	✓	✓	✓
XLA [64]	✗	✗	✗	✓	✓	✓
ONNX [65]	✗	✗	✗	✓	✓	✓
TVM [5]	✗	✗	✗	✓	✓	✗
EMC2 (**Ours**)	✓	✓	✓	✓	✓	✓

method MonoDETR [13] has a latency of 38ms with an average accuracies of 18.44%, and the stereo-based method ESGN [66] has a latency of 62ms on an NVIDIA RTX3090, with average accuracy 50.20%. For LiDAR-only methods, the performance is much better than camera-only methods even at similar latencies. The latencies of 3DSSD [39] on NVIDIA TITAN V, PartA2 [31] on NVIDIA TITAN Xp, and Voxel R-CNN [3] on NVIDIA 2080Ti GPU are 38ms, 71ms, and 40ms, respectively, with similiar accuracy around 55%. For LiDAR-camera fusion methods, although the performance increases compared to LiDAR-only methods, the latency also hugely increases due to the additional time needed for extracting image features. The latencies of RoboFusion [60] on NVIDIA A100 and GraphAlign++ [61] on V100 are 322ms and 149ms, respectively.

2.3 Apple-to-apple Object Detection Algorithm Comparison.

To better illustrate runtime performance on edge devices, we tested several common algorithms on a Jetson platform, with latency results shown in Tab. 3. The KITTI dataset divides sampled signals into three levels based on the number of pixels: Easy, Moderate, and Hard. To ensure high precision and low latency across these levels, we combined the low-latency advantage of LiDAR models with the high precision of LiDAR-plus-RGB models. Consequently, we designed a dynamic fusion acceleration framework to speed up the multimodal convolutional neural network. By flexibly adjusting matrix dimensions, the framework adapts to various edge computing devices, enabling high-precision recognition and low-latency computation with Camera and LiDAR inputs. A comparison of our framework's characteristics with existing frameworks is presented in Tab. 2.

2.4 Mixture of Expert

Mixture-of-Experts (MoE) [67] is a scalable neural network architecture that improves efficiency by dynamically activating specialized subnetworks (experts) based on input characteristics, thereby reducing computational overhead while maintaining capability. Modern MoE frameworks generally fall into three categories: general-purpose MoE frameworks, MoE-enhanced Transformer frameworks, and hardware-aware MoE frameworks.

General-Purpose MoE Frameworks. These frameworks integrate MoE across diverse deep-learning models. FastMoE [68] enables efficient expert parallelism in PyTorch while maintaining compatibility with standard libraries. FairScale MoE [69] extends FairScale to support distributed training and expert parallelism. DeepSpeed MoE [70] optimizes MoE training and inference for large-scale distributed setups.

MoE-Enhanced Transformer Frameworks. These frameworks incorporate MoE sparsity into Transformer models for better scalability and efficiency. Switch Transformer [71] scales parameter size via MoE layers while maintaining efficiency through sparse activation. GLaM [72] selectively activates experts per token, improving training efficiency. M6-T [73], an MoE-based Chinese NLP model, optimizes parameter utilization with MoE layers.

Hardware-Aware MoE Frameworks. These frameworks are optimized for specific hardware accelerators. Tutel [74] enhances expert parallelism and load balancing on GPUs and TPUs. Megatron-LM MoE [75] extends Megatron-LM for large-scale distributed training on NVIDIA GPUs, supporting tensor and expert parallelism.

While MoE enables efficient scaling of deep learning models, challenges remain, including communication overhead, load

Table 2. Comparison of the characteristics of existing frameworks with ours. For accuracy, a double checkmark indicates an average precision (AP) over 80, while a single checkmark represents an AP between 70 and 80. For latency, a checkmark indicates an inference time below 300 ms on standard GPUs.

Method	Easy		Moderate		Hard	
	Acc.	Latency	Acc.	Latency	Acc.	Latency
RGB	✓✓	✗	✓	✗	✗	✗
LiDAR	✓✓	✓	✓	✓	✗	✓
LiDAR+RGB	✓✓	✗	✓✓	✗	✓✓	✗
EMC2 **(Ours)**	✓✓	✓	✓✓	✓	✓✓	✓

imbalance, and training instability. Existing MoE frameworks mainly focus on training optimizations, leaving inference-time efficiency underexplored. Our work is the first to leverage MoE's flexibility for optimizing inference latency.

2.5 Machine Learning Compilation Platforms

Advanced ML compilers and deployment tools are crucial for efficiently utilizing various deep learning acceleration hardware. As ML models grow increasingly complex and computationally intensive, sophisticated deployment frameworks enable TensorFlow and PyTorch models to execute more efficiently across diverse hardware architectures. ML compilation platforms can be categorized into general-purpose compilation platforms, hardware-specific compilation platforms, and framework-specific compilation platforms.

General-Purpose Compilation Platforms. ONNX Runtime [76] provides an open standard for model interoperability, enabling seamless execution across CPUs, GPUs, and specialized accelerators. It enhances performance through computational graph and memory optimizations. Multilevel Intermediate Representation (MLIR) [77], built on LLVM, standardizes ML compiler pipelines, improving interoperability and scalability across frameworks and hardware. Apache TVM [5], a versatile deep learning compiler, supports automated tuning and tensor scheduling for efficient deployment across diverse hardware, including CPUs, GPUs, and FPGAs. It is particularly effective for cloud-based and edge AI inference and training.

Hardware-Specific Compilation Platforms. TensorRT [63], developed by NVIDIA, accelerates deep learning inference on NVIDIA GPUs using quantization, operator fusion, and layer optimizations, maximizing throughput while minimizing latency—ideal for real-time AI applications. OpenVINO [78], Intel's optimization toolkit, enhances deep learning performance on Intel CPUs, GPUs, and FPGAs through computation graph optimizations and model quantization. OneDNN [79] provides optimized deep learning primitives, leveraging CPU vectorization and memory optimizations for efficient inference execution.

Framework-Specific Compilation Platforms. Glow [80], an open-source compiler from Meta, optimizes PyTorch models through ahead-of-time compilation and quantization, enabling efficient execution on CPUs, GPUs, and accelerators—particularly in mobile and embedded environments. TorchDynamo [81], introduced in PyTorch 2.0, is a just-in-time (JIT) compiler that reduces Python overhead and integrates with backends like XLA [64], Inductor, and TensorRT to accelerate model execution.

3 Method

In ADS, different objects (e.g., pedestrians vs. trees) require varying levels of detection accuracy. Additionally, each sensing modality has distinct strengths and limitations: LiDAR offers detailed 3D spatial information but suffers from sparsity due to limited laser reflection intensity, whereas camera offers fine-grained textures but introduce redundancy, making computation expensive. Thus, a one-size-fits-all detection pipeline struggles to maintain both latency and accuracy across diverse scenarios while fully leveraging each modality's strengths.

To tackle these challenges, we propose EMC2, a multimodal MoE framework with an adaptive expert switching mechanism that balances latency and accuracy in autonomous driving perception. As illustrated in Fig. 2, EMC2 consists of four key components: (1) Adaptive Multimodal Data Bridge (Sec. 3.1), which preprocesses multimodal inputs for different experts; (2) MoE Switch (Sec. 3.2), which dynamically assigns the most suitable expert based on scene characteristics; (3) Latency-

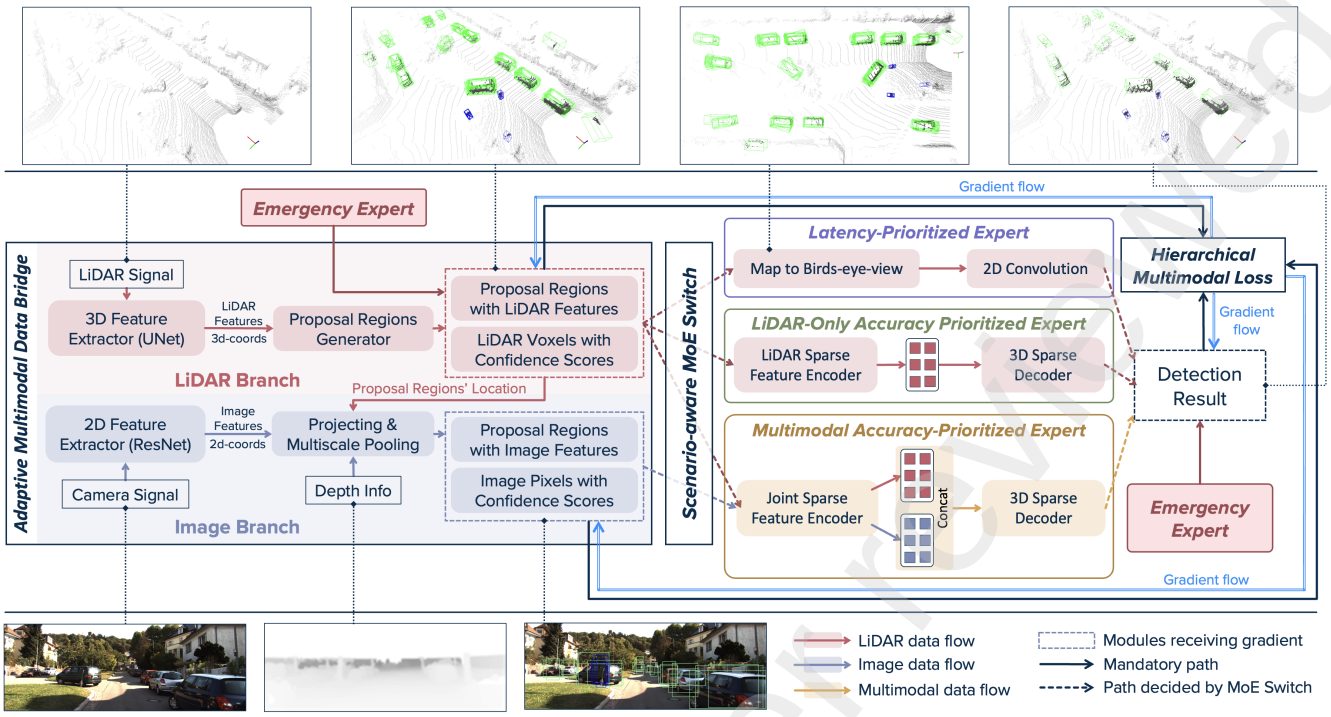


Figure 2. Illustration of the EMC2 **Framework.** In the figure, the middle row represents the EMC2 architecture, while the top and bottom rows show visualizations of selected modules and the legend. The system consists of four key components: (1) an *Adaptive Multimodal Data Bridge (AMDB)* that preprocesses raw sensor data and provides features tailored to each expert's needs; (2) an *MoE Switch* that dynamically selects experts based on the scenario, as illustrated in Fig. 3; (3) a *Latency-Prioritized Expert* for handling straightforward scenarios where low latency is preferred; (4) an *Accuracy-Prioritized Expert* for complex scenes requiring high precision. An *Emergency Expert API* is considered for responding to high-risk, unseen scenarios. The loss function incorporates both detection results and intermediate outputs from the *AMDB*, as detailed in Sec. 3.5.

Prioritized Expert (Sec. 3.3) and (4) Accuracy-Prioritized Expert (Sec. 3.4), which decode features into object bounding boxes and classification confidences. All experts operate in parallel, but MoE dynamically selects and activates only the most suitable expert per scenario. To ensure robust expert training, we introduce Hierarchical Multimodal Loss and Back-Propagation (Sec. 3.5).

For edge-device deployment, we leverage ONNX Runtime with CUDA for framework compilation. We develop optimized 3D Sparse Convolution and Multiscale Pooling layers (Sec. 3.6) within ONNX tailored to edge scenarios. Coupled with Hierarchical Training, these algorithmic optimizations maximize efficiency without sacrificing performance. Additionally, we perform system-level optimizations, including memory management and computation graph refinement (Sec. 3.7), that further amplify algorithmic improvements and optimize ONNX Runtime execution.

3.1 Adaptive Multimodal Data Bridge

LiDAR provides 3D spatial information but often misses low-reflectivity areas, resulting in sparse sampling. Image captures rich details but also contains high redundancy, driving up computational cost. To balance these trade-offs, we propose an *Adaptive Multimodal Data Bridge (AMDB)* that uses LiDAR data for preliminary image segmentation, then augments specified voxels in those segmented regions with image features. This strategy provides enough information for downstream experts without incurring excessive computation.

The workflow of *AMDB* starts by employing a UNet and sparse convolutional neural network (CNN) to extract LiDAR features, then uses fully connected layers to generate proposal regions and their confidence scores. These intermediate results

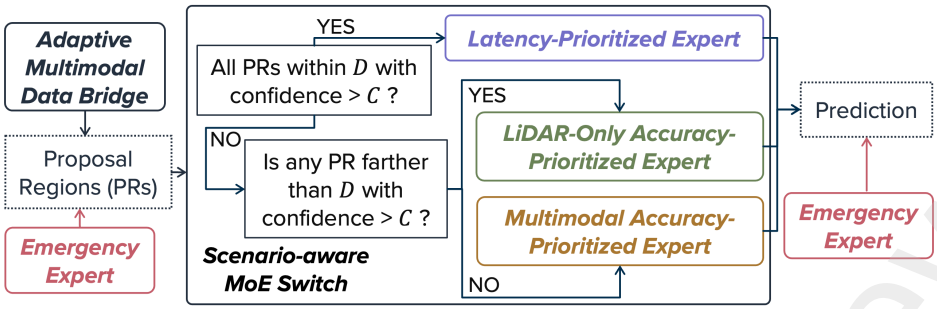


Figure 3. Routing Strategy of MoE Switch. Safety distance D and Confidence score threshold C are adjustable. For unexpected events, we designed an Emergency Expert API for adapting to risky, unseen scenarios, triggered by erratic proposal regions given by *AMDB* and final predictions given by experts, enabling immediate braking or evasive maneuvers without passing through the entire EMC2. This design increases emergency sampling frequency and reduces response time.

will be used by *Latency-Prioritized Expert* and *Accuracy-Prioritized Expert* later. The extraction of image features is depend on *MoE Switch*. If the multimodal mode is activated, *AMDB* will extracts image features via ResNet. By incorporating depth, it projects the relevant image pixels into 3D space, applies a Multiscale Pooling operation to reduce computational overhead, and fuses them with LiDAR voxel features to produce a multimodal output, illustrated as Alg. 1. The Multiscale Pooling and Multimodal Fusion processes are illustrated intuitively in Fig. 5.

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

3.2 MoE Switch

To balance latency and accuracy across diverse driving scenarios, 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. 3. By training experts specialized for different conditions, and routing each scenario to the appropriate one, the *MoE Switch* significantly reduces average inference time while maintaining accuracy within safety thresholds.

During inference, we employ the MoE routing strategy shown in Fig. 3. Trained experts are selected based on object distance and clarity, where clarity is inferred from confidence scores of proposal regions provided by the *AMDB*. More specifically, the *Latency-Prioritized Expert* handles scenes where all objects are clearly visible and no distant objects are present, avoiding false positives in far ranges. The *LiDAR-Only Accuracy-Prioritized Expert* processes scenes containing unclear objects but no distant objects, or scenes with only clear objects that include distant objects. After the R test, it is found that there is a 95.62% probability that distance and object clarity are related. So in this work, the *Multimodal Accuracy-Prioritized Expert* is responsible for scenes with both unclear objects and distant objects. These experts will be described in detail in the following

section.

3.3 Latency-Prioritized Expert

To efficiently handle near-field objects with dense and well-defined voxel information, we design the *Latency-Prioritized Expert (LPE)* based on 2D CNNs, prioritizing inference speed while maintaining reliable detection accuracy. Compared to traditional 2D CNN operations, 3D CNN introduce additional time complexity of $O(T \times K_T)$, where K_T represents the kernel size along the temporal dimension, and T denotes the third dimension (height in our case). In scenarios with abundant voxel information, even when compressed into Bird's Eye View (BEV) space, 3D CNN can still accurately capture height, preventing false positives and missed detections.

The *LPE* firstly projects proposal regions in 3D voxel-space, produced by *AMDB*, into BEV representation. It then extracts features and performs object localization and classification using a 2D CNN, before mapping the 2D CNN detection results back into 3D space. The experimental results are presented in Sec. 4.

3.4 Accuracy-Prioritized Expert

Accuracy-Prioritized Expert (APE) is designed for scenarios with far-field objects and significant missing voxel information. Two configurations are offered: *LiDAR-only APE* and *Multimodal APE*. *LiDAR-only APE* applies a 3D sparse CNN to process proposal regions and confidence scores from *AMDB*, and generate per-voxel classification and bounding boxes via the rest part of the trained decoder after the 3D spare CNN. *Multimodal APE* processes multimodal proposal regions (also from *AMDB*) using a different decoder, integrating LiDAR and camera data to compensate for missing LiDAR details, significantly improving performance on distant and challenging objects. Sec. 4 presents experimental details and results for both configurations.

3.5 Hierarchical Training and Long-Tail Mitigation

Hierarchical Multimodal Loss and Back-Propagation. During training, since the experts rely heavily on proposal regions generated by the *AMDB*, yet these proposal regions are unreliable in early epochs, this significantly affects the decision-making of the *MoE Switch* and introduces noise into expert training. To mitigate this, we implement Hierarchical Back-Propagation, as shown in Fig. 4, comprising three separate back-propagation routes to supervise: (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 results and confidence scores from experts. Moreover, we pre-train the two branches of the *AMDB* separately to stabilize their outputs before joint training. Each back-propagation route is supervised by the following loss function, which collectively forms our Hierarchical Multimodal Loss:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{cls}} + \frac{\mathcal{L}_{\text{reg}}}{N_{\text{fore}}} \tag{1}$$

where N_{fore} represents foreground voxels associated with objects to be detected, as detailed in Sec. 4.1. The classification loss $\mathcal{L}_{\text{cls}}$ and regression loss $\mathcal{L}_{\text{reg}}$ are formulated as:

$$\mathcal{L}_{\text{cls}} = \mathcal{L}_{\text{ce}}(y_c, \hat{y}_c) \tag{2}$$

$$\mathcal{L}_{\text{reg}} = \mathcal{L}_{\text{smooth-L1}}(y_r, \hat{y}_r) \tag{3}$$

where $\hat{y}_c$ and $\hat{y}_r$ are the predicted class and location of proposal regions (if supervising *AMDB*) or final detection results (if supervising experts), y_c and y_r are their ground-truth values, $\mathcal{L}$ce denotes binary cross-entropy loss for classification, and $\mathcal{L}$smooth-L1 is smooth-L1 loss for localization. This hierarchical training design not only stabilizes expert learning but also enhances the model's adaptability to varying data granularities. As a result, even under compressed data representations, it enables the effective use of our Multiscale Pooling for memory efficiency, as detailed in Sec. 3.6.

Addressing Long-tail Effects in MoE Training. In traditional MoE frameworks [67], long-tail effects arise when certain experts receive insufficient data due to imbalanced training distributions or skewed expert-allocation policies [6], leading to

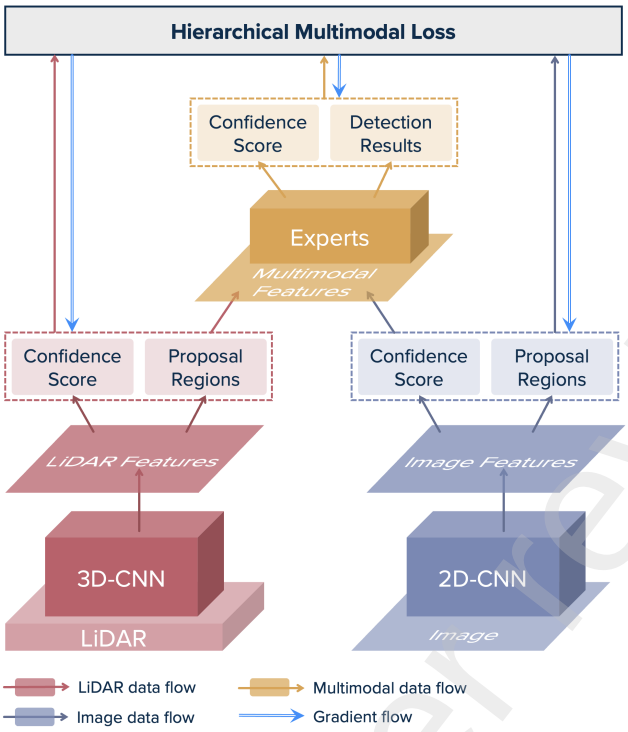


Figure 4. Hierarchical Multimodal Loss and Back-Propagation of EMC2**.** The three paths include: (1) backpropagation from the final detection output, (2) backpropagation from the LiDAR branch of *AMDB*, and (3) backpropagation from the image branch of *AMDB*. Since expert training heavily depends on 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.

reduced performance and inefficient resource usage. To mitigate this, we firstly propose a data subset division method that partitions the data into subsets based on target T_{C_i} and interference I_{C_i}. For K experts $E = \{E_1, E_2, \ldots, E_K\}$, there are K data subset with each subset denoted as C_i, and each E_i is assigned a subset C_i. Each subset includes the target T_{C_i} and interference I_{C_i}. To enhance each expert's ability to suppress interference, we ensure that $C_i \cap C_j = I_{C_{i,j}}$ and $T_{C_i} \cap T_{C_j} = \emptyset$. Secondly, we adjust the selection probability λ_i for subsets with fewer samples, such that $\lambda_i \times N_{C_i} = \lambda_j \times N_{C_j}$ and $\sum_{i=1}^{K} \lambda_i = 1$. This sampling strategy that experts for less frequent classes receive sufficient training examples. Furthermore, to mitigate the impact of interference within each expert's data subset, we introduce regularization terms into the loss functions supervising each expert. These terms encourage the model to focus on its designated target classes while suppressing misleading gradients from interfering classes:

$$\mathcal{L}_{\text{exp-cls}} = \mathcal{L}_{\text{ce}}(y_c, \hat{y}_c) + \mathcal{R}_{\text{cls}} \tag{4}$$

$$\mathcal{L}_{\text{exp-reg}} = \mathcal{L}_{\text{smooth-L1}}(y_r, \hat{y}_r) + \mathcal{R}_{\text{reg}} \tag{5}$$

where $\mathcal{L}_{\text{exp-cls}}$ and $\mathcal{L}_{\text{exp-reg}}$ correspond to classification and regression for final detection results, respectively, with other terms defined in Eq. 2 and Eq. 3. The regularization terms $\mathcal{R}_{\text{cls}}$ and $\mathcal{R}_{\text{reg}}$ are defined as follows:

$$\mathcal{R}_{\text{cls}} = -\alpha(1 - p_t)^{\gamma} \log(p_t) \tag{6}$$

$$\mathcal{R}_{\text{reg}} = \sum_{i} \lambda (\hat{y}_i - y_i)^2 \tag{7}$$

where p_t represents the probability of interference presence, γ adjusts the emphasis on hard samples, α balances the contribution of different samples, and λ is a weighting hyper-parameter. These parameters are detailed in Sec. 4.1.

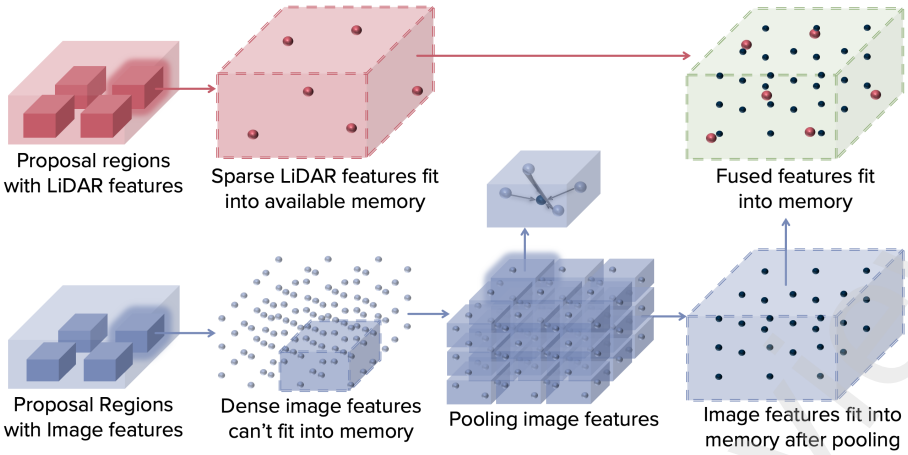


Figure 5. Demonstration of Multiscale Pooling and Multimodal Fusion. Proposal regions with image features are obtained by projecting the image pixels—those lying within proposal regions generated by *AMDB* using LiDAR features—into 3D. The pooling and fusion steps begin after this projection. Once pooling is applied, any pixel sharing the same 3D coordinate as a voxel has its image features concatenated with the voxel's LiDAR features, while others have them concatenated with zero-filled features (Alg. 1).

3.6 Algorithmic Enhancement for Edge Computing Efficiency

Edge computing devices have limited cache, longer memory access latency, and incomplete function libraries, making efficient data and memory management crucial. Additionally, 3D convolution operations dominate the inference runtime. To address these challenges, at the algorithm level, we employ optimized 3D Sparse Convolution [82] and design Multiscale Pooling to optimize multimodal feature fusion processing while reducing computational overhead.

3D Sparse CNN. 3D sparse convolution have a computational complexity of $O(\mathcal{N} \times C_{\text{in}} \times \mathcal{K}^3 \times C_{\text{out}})$, whereas dense convolutions operate at $O(\mathcal{H}^3 \times C_{\text{in}} \times \mathcal{K}^3 \times C_{\text{out}})$, where $\mathcal{N}$ is the number of nonzero voxels, C_{in} and C_{out} are the input and output channels, $\mathcal{K}^3$ is the kernel size, and $\mathcal{H}^3$ is the total spatial domain. By restricting computation to nonzero voxels, sparse convolution significantly reduces redundant operations, lowering computational overhead by 65–80% [36]. To make it more suitable for edge devices, we developed a customized Sparse 3D Convolution library with parallel execution and Fused Multiply-Add (FMA) operations that fits into ONNX Runtime compilation platform, as illustrated in Alg. 2. These implementations are enhanced through system-level optimizations, which are detailed in Sec. 3.7.

Algorithm 2 Optimized 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 $\mathcal{V}_{\text{out}} = \{f_{\text{out}}^i, o_{\text{out}}^i\}_{i=1}^N$.

for $v_{\text{in}}^i = \{f_{\text{in}}^i, o_{\text{in}}^i\} \in \mathcal{V}_{\text{in}}$ **do in parallel**
  $(\mathrm{x}_{k_0}, \mathrm{y}_{k_0}, \mathrm{z}_{k_0}) \leftarrow o_{\text{in}}^i$ *# Align kernel center to voxel*

  # Collect all voxels currently covered by kernel
  $\mathcal{V}_{\text{part}} \leftarrow \{(f_{\text{in}}^j, o_{\text{in}}^j) \mid o_{\text{in}}^j = (\mathrm{x}_c + \mathrm{kx}, \mathrm{y}_c + \mathrm{ky}, \mathrm{z}_c + \mathrm{kz}), \forall (\mathrm{kx}, \mathrm{ky}, \mathrm{kz}) \in \mathcal{K}, o_{\text{in}}^j \in \mathcal{V}_{\text{in}}, o_{\text{in}}^j \in \mathcal{H}\}$

  # Perform Fused Multiply-Add (FMA) as convolution
  $(f_{\text{out}}^i, o_{\text{out}}^i) \leftarrow \textbf{FMA}(\mathcal{V}_{\text{part}}, \mathcal{K})$
end for

Multiscale Pooling. In multimodal data processing, the combined voxel count from LiDAR and image features often exceeds the cache capacity of edge devices, constraining performance. As illustrated in Fig. 5, we propose Multiscale Pooling, which adaptively pools image feature voxels based on available cache and then fuses them with LiDAR feature voxels, achieving

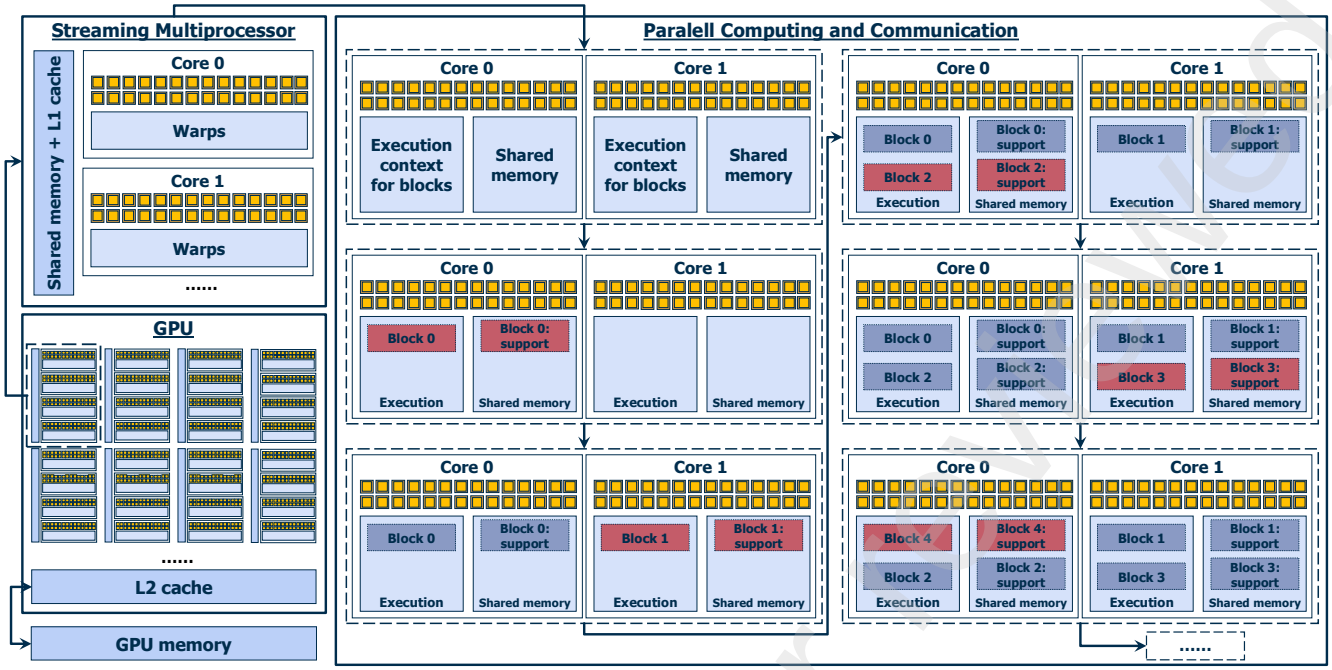


Figure 6. Illustration of Streaming Multiprocessor (SM), Shared Memory, and Parallel Operations. SMs form the core computational units of a GPU, each containing cores for parallel execution. When a kernel is launched, it defines computation blocks that will be distributed across SMs. Shared memory provides 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.

a balance between computational efficiency and effective feature complementarity. The pooling size is user-configurable, allowing adaptability to different hardware constraints. Coupled with our Hierarchical Back-Propagation strategy (Sec.3.5), Multiscale Pooling significantly reduces resource consumption while maintaining model reliability, as demonstrated in Sec. 4.4.

3.7 System-Level Optimization for Edge Deployment

To further amplify the algorithmic optimizations in Sec. 3.6, at system level, we implement memory and computation graph optimization to streamline execution, ensuring efficient deployment across diverse hardware platforms.

Memory Optimization. We propose three strategies to accelerate the runtime of memory usage, including: (1) Reducing global memory access; (2) Parallel communication and computation and (3) Prefix Scanning for sparse CNN.

Reducing global memory access [83]. Shared memory, shown in Fig. 6, is a high-speed cache within each Streaming Multiprocessor (SM) that reduces global memory access by enabling data sharing among cores. Optimizing its usage greatly boosts GPU efficiency and minimizes latency. To further enhance this, this work set the ONNX compiler to maximize shared memory hit rates and optimize execution order across warps, minimizing redundant memory access, and eventually improving overall computational performance.

Parallel Communication and Computation [84]. A naive approach would load all data to the GPU before any computation begins. This achievement restricts the GPU can only computation or communication at a given time. To fully utilize GPU computation and communication resources, we partitions matrices into multiple chunks. While one chunk is being processed, the subsequent chunk is loaded into memory and the previous chunk is offloaded, allowing concurrent computation and communication to enhance efficiency, as shown in Fig. 7.

Thread Management [85]. To enhance hardware efficiency and optimize memory utilization, we designed a four-stage thread management strategy. After each stage, the CPU terminates all active threads, triggers a system interrupt, and launches a new

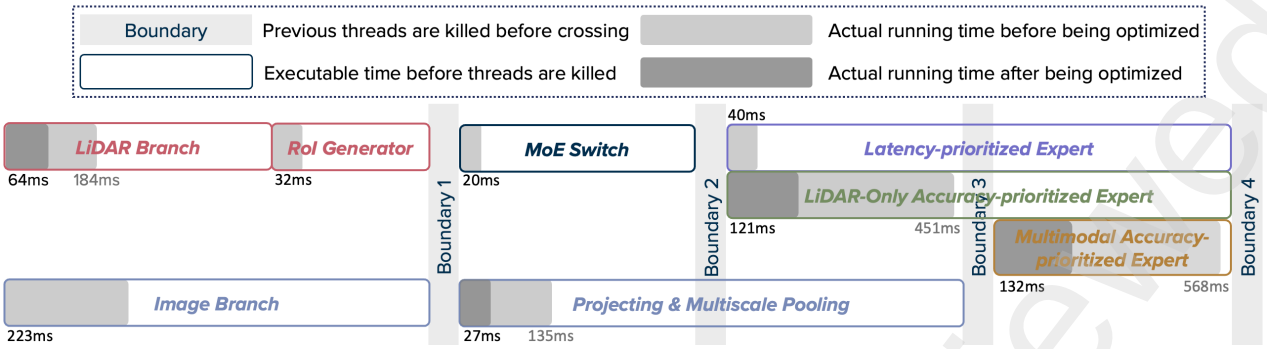


Figure 7. Illustration of Thread Management. We define four boundaries to manage all threads in EMC2. Upon reaching each boundary, the preceding threads are terminated, and their allocated address space is released. Each component spans a set of threads, with light gray strips and numbers indicating execution time before optimization, and darker strips and numbers representing execution time after optimization.

batch of threads for the next stage. This approach ensures that, at any given stage, memory only retains the intermediate state of the current stage, eliminating the need to store intermediate states from other stages. As a result, memory utilization is significantly improved at the system level. The specific management strategy is illustrated in Fig. 7.

Prefix scanning for sparse CNN operation [86]. To further optimize hardware management, we leverage GPU-based prefix scan capabilities. This parallel technique (often used to compute cumulative sums or products) reduces the number of certain calculation time steps from $O(N)$ to $O(logN)$, improving load balancing and overall hardware efficiency.

Computation Graph Optimization. Our ONNX computation graph optimization occurs in two stages [65]: (1) Graph rewriting (before partitioning on each streaming multiprocessor subcore) and (2) Complex node fusion (after partitioning).

Graph Rewriting. Graph rewriting includes constant folding, redundant node elimination, and semantic-preserving node fusion. Constant folding removes computations dependent on constants, reducing runtime overhead. Redundant node elimination discards unnecessary operations without altering the graph's structure. Semantic-preserving fusion merges multiple operators (e.g., Conv + Add) into a single node.

Complex Node Fusion. After partitioning, complex node fusion further optimizes execution efficiency. We use General Matrix Multiply (GEMM) activation fusion that integrates polynomial computations with activation functions, while addition-multiplication fusion reduces separate arithmetic operations. Layer normalization fusion condenses multi-step operations (e.g., ReduceMean, Sub, Pow, Sqrt, Div) into a single LayerNormalization operation. Lastly, expressions like LayerNorm$(X + f(X))$ are fused into a single operation.

4 Experiments

To better reflect real-world autonomous driving scenarios, we conduct our experiments on the Jetson AGX Orin, a resource-constrained edge computing platform with only three cache levels and a 4MB L2 cache per GPU core. Without advanced sparse convolution libraries, matrix operations exceeding this limit incur significant memory swapping overhead, impacting efficiency. We evaluate EMC2 on the KITTI dataset for 3D object detection, using the standard split of 3,712 training and 3,769 validation samples. Performance is measured using the widely-used protocol of 3D Average Precision (AP) at 40 recall thresholds (R40), with IoU thresholds of 0.7 for cars and 0.5 for pedestrians and cyclists, applied uniformly across all three dimensions of the 3D bounding box. Importantly, we also consider inference time on embedded platforms as a key performance metric in addition to accuracy.

4.1 Implementation Details

Data Sampling Details for Addressing Long-tail Effect. In Sec. 3.5, we discussed using a resampling method to address the long-tail effect. In the KITTI dataset, car annotations make up about 60-70% of all labels, the remaining 30-40% includes

annotations for cyclists and pedestrians, with cyclists being less frequently labeled than pedestrians. Regarding difficulty levels, about 30% of labels are classified as “easy”, 40% as “moderate,” and 30% as “hard”. Essentially, the difficulty is determined by factors such as object size in voxels and degree of occlusion. Based on the characteristics of *LPE*, we set it up with a targeted subset T_{C_1} containing scenes within all objects in 30 meters and only contains “easy” or “moderate” level objects. Similarly, we configured I_{C_1} containing scenes that include "hard" level objects in 30 meters, or include objects out of 30 meters. I_{C_1} matching the data quantity in T_{C_1}. As for *APE*, we defined the target subsets T_{C_2} and T_{C_3} for its LiDAR and Multimodal configuration respectively as the complement of T_{C_1}, while I_{C_2} and I_{C_3} serve as complements of I_{C_1}, with data quantity approximately equal to that in T_{C_2} and T_{C_3}.

Hierarchical Loss Details. To ensure effective joint training of each Expert and *AMDB*, we design the loss function as shown in Eq. 1. A higher N_{fore} indicates more raw voxels associated with Cars, Pedestrians, or Cyclists. In the KITTI dataset, objects other than these three types (e.g., Trucks) are ignored in evaluation but still appear in the data. If mistakenly treated as valid objects, these ignored objects could cause unintended accelerations in gradient descent. To mitigate this, we leverage the N_{fore} term to initialize the regression characteristics of the $\mathcal{L}_{\text{reg}}$ loss.

Details of Regularization Terms for Classification. The $\mathcal{R}_{\text{cls}}$ term in the classification loss $\mathcal{L}_{\text{cls}}$ is related with p_t in Eq. 6, which represents the probability of interference. For different experts, we set:

$$p_t = \frac{T_{C_i}}{T_{C_i} + I_{C_i}}, \tag{8}$$

T_{Ci} refers to the count of target training samples for expert E_i. I_{C_i} refers to the count of interfering samples belonging to other categories for expert E_i. To ensure the convergence of the gradient descent, both p_t is constrained within 0 and 1. When there is only one expert in the framework, we have

$$p_t = \frac{T_{C_i}}{T_{C_i} + I_{C_i}} = 1, \tag{9}$$

and in turn we have:

$$\mathcal{R}_{\text{cls}} = 0, \tag{10}$$

In the one expert case, we do not need to consider suppressing long-tail effects. γ in $[0, 1]$ could help to ensure a convergent gradient descent process, and its value varies depending on the sample. For example, in I_{C_i} scenarios we use $\gamma = 0.3$, while in T_{Ci} scenarios we use $\gamma = 0.7$. Since batch size during model training is not always one, when a batch contains both T_{Ci} and I_{C_i} scenes, the value of γ is computed using:

$$\gamma = \sigma \cdot N_{T_{Ci}} + (1 - \sigma) \cdot N_{I_{C_i}}, \tag{11}$$

where we set $\sigma = 0.7$, $N_{T_{C_i}}$ is the number of scenes T_{C_i} in this batch, and $N_{I_{C_i}}$ is the number of scenes I_{C_i} in this batch. To maintain a stable gradient descent process, the parameter α in Eq. 6 dynamically changes with $\mathcal{L}_{\text{cls}}$ in Eq. 2.

Details of Regularization Terms for Regression. The $\mathcal{R}_{\text{reg}}$ term in Eq. 7 represents an L_2 loss. We observed that using a smooth loss function often leads to overfitting [31], particularly for distant objects with sparse LiDAR reflections. In such cases, the lack of dense LiDAR signals causes smooth loss-based training to produce excessive false positives. To mitigate this, we incorporate $\mathcal{R}_{\text{reg}}$ into the $\mathcal{L}_{\text{reg}}$ functions of both the *AMDB* and the Experts, as formulated in Eq. 3. This adjustment enables the *AMDB* to generate more reliable more focused proposal regions while allowing the Experts to refine final predictions, thereby improving recall.

4.2 Result Visualization

Fig. 8 presents the detection results of PV-RCNN [87] and Second [36] as baselines, alongside our proposed EMC2, on four representative scenes from the KITTI validation dataset.

For objects closer to the LiDAR sensor, stronger reflection intensity results in a higher number of valid voxels and higher confidence scores. As distance increases, reflection weakens [88], LiDAR points become sparser, and confidence scores decline.

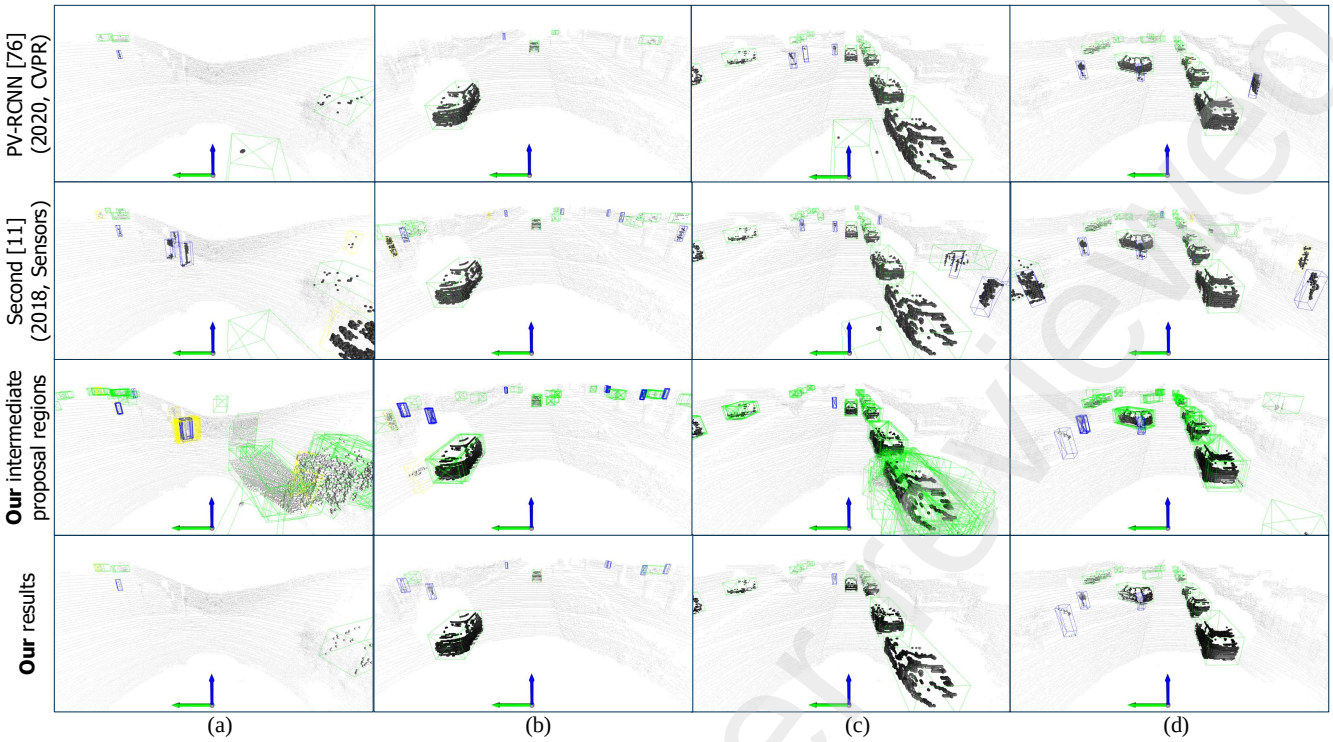


Figure 8. Visualization of detection result of two baselines and our approach. We select four representative scenarios: (a) intersection, (b) standard, (c) narrow street, and (d) dense surrounding. The first two rows show the detection results of PV-RCNN [87] and SECOND [36], respectively. The third row illustrates the proposal regions generated by our *AMDB*, while the fourth row presents the final detection results of EMC2. 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, EMC2 significantly reduces both false positive and false negative detections.

For vehicles, reflections from windows and rooftops fade the fastest, disappearing beyond 60 meters, while signals from doors weaken gradually and vanish after 90 meters. The vehicle body and tire framework remain detectable within 120 meters but disappear beyond this range [88]. Vehicles moving in the opposite direction tend to have lower scores for windows and rooftops due to smaller LiDAR reflection angles. Pedestrians and bicycles exhibit more variability. At the same distance, confidence scores of voxels for pedestrians can differ significantly, largely due to variations in clothing materials [88]. In contrast, bicycles generally maintain consistent confidence scores across different parts at the same distance, though all scores decline uniformly with increasing distance. Regarding occlusion, the since LiDAR sensor is roog-mounted [89], it captures substantial information from vehicles in the same lane. As demonstrated in Fig. 8, pedestrians, vehicles, and bicycles are effectively detected because their LiDAR signals differ significantly in height and spatial distribution from background points. This enables 3D sparse convolution to extract spatial features like distance and height, regardless the interference from ground clutter.

As shown in Fig. 8, compared to EMC2, PV-RCNN [38] exhibits an additional false positive and false negative for pedestrians in scene (a), along with one false negative and three false positives for cars. In scene (b), it introduces one additional false positive for cars. In scene (c), while it has two fewer false positives for cars, it also produces four more false negatives for pedestrians. In scene (d), there are two additional false positives for pedestrians. Similarly, compared to EMC2, Second [36] in scene (a) generates two extra false positives for both cars and pedestrians. In scene (b), it adds three false positives each for bicycles, pedestrians, and cars. In scene (c), it produces three more false positives for cars, one for pedestrians, and three for bicycles. In scene (d), it has two fewer false negatives for pedestrians but introduces three additional false positives for cars and

Table 3. Evaluation of different methods for 3D object detection on KITTI dataset using various metrics and computation times. Some spots are marked as N/A because the open-source version of the respective model does not support that particular class, or as +∞ because the respective model does not support Jetson deployment.

Method	Pedestrian 3D AP (R40)			Car 3D AP (R40)			Cyclist 3D AP (R40)			Inference Time	
	Easy	Moderate	Hard	Easy	Moderate	Hard	Easy	Moderate	Hard	A4000	Jetson
LiDAR-only Models											
M3DETR [90]	69.69	66.04	61.61	93.96	86.28	84.32	87.88	72.47	70.63	338	+∞
PV-RCNN [87]	N/A	N/A	N/A	91.18	87.65	83.14	N/A	N/A	N/A	45	1787
Voxel-RCNN [3]	N/A	N/A	N/A	92.23	85.04	82.50	N/A	N/A	N/A	70	+∞
GLENet-VR [91]	N/A	N/A	N/A	93.51	86.22	83.72	N/A	N/A	N/A	165	+∞
PartA2-Anchor [62]	66.87	59.68	54.60	92.15	82.92	82.10	90.34	70.05	66.89	95	+∞
PartA2-Free [62]	72.31	66.36	60.06	91.66	80.28	78.08	91.88	75.33	70.67	124	1064
PointPillar [32]	57.29	51.41	46.87	87.75	78.40	75.19	81.57	62.93	58.97	41	972
Second [36]	55.94	51.14	46.16	90.55	81.60	78.60	82.96	66.73	62.78	45	1322
Second-IoU [36]	61.10	54.66	49.50	91.53	82.36	79.62	90.73	71.23	66.26	58	+∞
3ONet [92]	72.55	65.21	60.22	**94.24**	87.32	84.17	92.47	75.11	71.18	100	+∞
Multimodal Models											
Voxel-RCNN [3]	N/A	N/A	N/A	92.08	85.90	83.36	N/A	N/A	N/A	331	965
LoGoNet [93]	70.02	63.72	59.46	92.04	85.04	84.31	91.74	75.35	72.42	100	+∞
VFF [94]	73.26	65.11	60.03	92.24	85.51	82.92	91.74	75.35	69.84	192	+∞
CAT-DET [95]	74.08	66.35	58.92	90.12	81.46	79.15	87.64	72.82	68.20	154	+∞
TED [96]	74.73	69.07	63.63	92.25	88.94	86.73	**95.20**	76.17	71.59	100	+∞
EMC2 **(Ours)**	**74.92**	**69.92**	**66.81**	94.00	**89.35**	**88.15**	95.06	**79.87**	**77.62**	162	**372.5**

one more for pedestrians.

4.3 Comparison with Other SOTA Methods

In Tab. 3, we present the comparative results of our EMC2 against SOTA 3D detection methods on the KITTI validation set under the standard split. Notably, for pedestrian and cyclist detection across all difficulty levels, our model achieved substantial accuracy gains, outperforming existing SOTA models by an absolute margin of 1.42–6.03 percentage points while reducing total error by 5–10%. Pedestrian and cyclist detection plays a crucial role in ensuring safety, and our model effectively addresses this need. For vehicle detection, our model achieves over 88.15% accuracy across all difficulty levels, meeting current safety-based standards.

We evaluated our model's runtime performance on the Jetson platform using both PyTorch and C frameworks. Specifically, our *LPE* achieves an average runtime of 219 ms and *APE* 526 ms, resulting in an overall average of 372.5 ms across experts. In contrast, other SOTA models, which rely on 2D dense convolutions in Backbone-2D, require 971–1787.2 ms when using mixed C/Python frameworks. The PointPillar model, compiled into an ONNX file under the C framework, runs at 40 ms on the Jetson AGX Orin. However, despite its speed, its accuracy drops by 0.2–0.5% for vehicles and 14–15% for pedestrians, making it insufficient to meet safety standards.

In summary, our work establishes a new balance between safety, accuracy, and accelerated execution, providing a robust solution for real-time 3D detection in autonomous driving.

4.4 Ablation Study

Impact of Latency-Prioritized Expert. We explore how *LPE* improves efficiency without compromising detection reliability compared to *APE* in straightforward scenarios. We evaluated three configurations: (1) the *LPE*, (2) the *LiDAR-Only APE*, and (3) the *Multimodal APE*, using the validation subset assigned to *LPE*. The results, presented in Tab. 4, show minor performance differences among the three, while *LPE* achieves significantly faster inference speeds on the Jetson platform. This validates that deploying *LPE* in appropriate scenarios is a sensible choice, as it reduces latency while maintaining accuracy above an acceptable safety limit.

Impact of Multimodal Accuracy-Prioritized Expert. We compared the performance of the *APE* in LiDAR-Only and

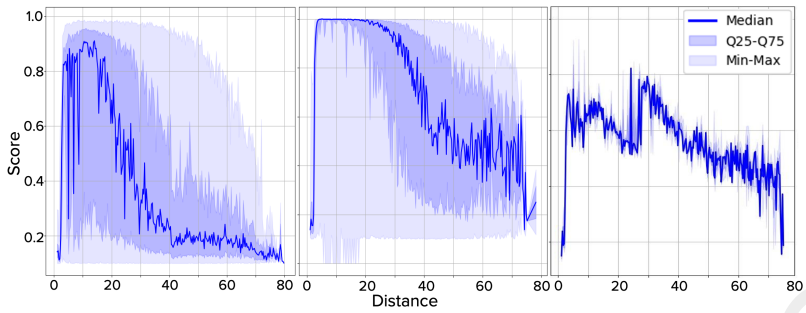
Table 4. Comparison of accuracy and latency across three controlled trials in the ablation study. The main body of the table is divided into three sections by double lines, providing support for the three parts of the ablation study.

Method	Accuracy ↑			Latency (ms) ↓
	Pedestrain	Car	Cyclist	
Impact of Latency-Prioritized Expert				
LPE	66.32	89.04	88.54	**219**
LiDAR-Only APE	**74.49**	92.23	91.69	468
Multimodal APE	74.34	**93.21**	**92.41**	526
Impact of Multimodal Accuracy-Prioritized Expert				
LiDAR-Only APE	67.24	87.69	81.85	**468**
Multimodal APE	**70.55**	**90.5**	**84.18**	526
Impact of Hierarchical Training				
HT ✓, MP ✓	**70.55**	**90.50**	**84.18**	**526**
HT ✓, MP ✗	**70.55**	**90.50**	84.07	1320
HT ✗, MP ✓	63.11	88.68	75.98	529
HT ✗, MP ✗	66.34	89.1	80.83	1367

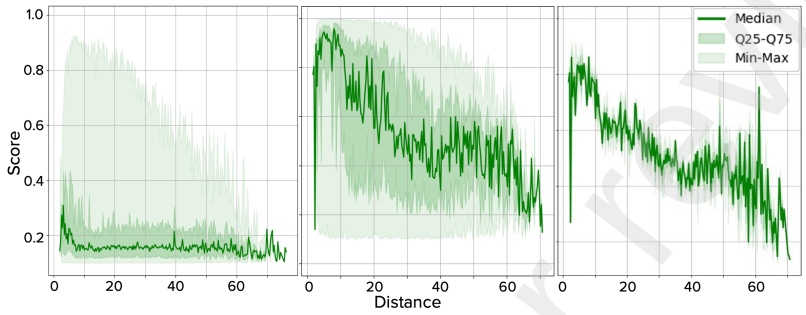
Multimodal configurations on the full KITTI validation set to evaluate the impact of incorporating the Image branch from the *AMDB*. The results, shown in Tab. 4, indicate a significant improvement in average detection performance, with a slight increase in inference time on the Jetson platform. Notably, when LiDAR reflections weaken due to long distances or challenging angles, point cloud data for certain objects becomes highly sparse—especially in hard case samples. By projecting encoded pixel features from 2D image space into 3D, we effectively compensate for missing information in sparse LiDAR data, enhancing detection robustness and reliability.

Impact of Hierarchical Training and Multiscale Pooling. We mentioned in Sec.3.5 and Sec.3.6 that implementing Hierarchical Training supervises the three branches of EMC2, ensuring that intermediate results provided by the *AMDB* are reliable and stabilizing expert training. This training strategy significantly enhances the model's ability to extract features at different levels of granularity, making Multiscale Pooling feasible for efficiency purpose. Here, we evaluate the impact of this setup through four configurations: (1) training with Hierarchical Training and inference with Multiscale Pooling; (2) training with Hierarchical Training but inference without Multiscale Pooling; (3) training without Hierarchical Training but inference with Multiscale Pooling; and (4) training without Hierarchical Training and inference without Multiscale Pooling. All configurations were trained for the same number of epochs and evaluated on the full KITTI validation set. The results are shown in Tab. 4, where HT represents Hierarchical Training, and MP represents Multiscale Pooling. The results indicate that while accuracy remains consistent between the first two configurations, Multiscale Pooling significantly reduces inference time. In contrast, for models trained without Hierarchical Training, accuracy deteriorates. Besides, inference time increases drastically in configurations without Multiscale Pooling. These results validate that Hierarchical Training plays a critical role in feature extracting, especially when combined with Multiscale Pooling, which effectively accelerates inference without compromising performance.

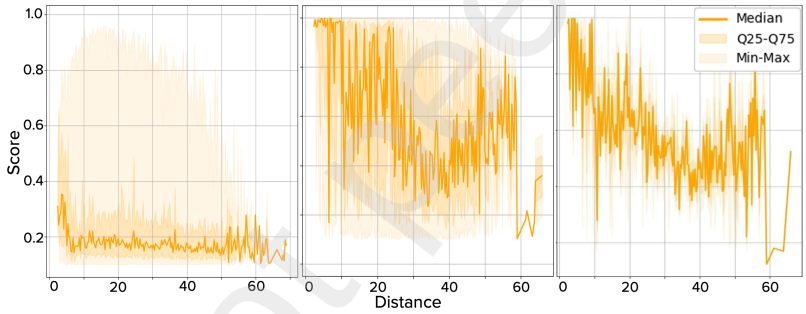
The Relationship Between Detection Confidence Score and Object Distance. Based on each expert's performance across different scenarios, we observed that an expert's accuracy depends not only on object difficulty but also on object distance. To analyze this, we visualize the relationship between detection confidence score and object distance under three configurations: (1) the *LPE*, (2) the *APE*, and (3) the full EMC2 system, as shown in Fig. 9. At close range, both the *LPE* and the *APE* achieve consistently high confidence scores across all object types, statistically justifying the use of the *LPE* for near-field objects, which are easier to recognize. As distance increases, the confidence scores of the *LPE* decline significantly, whereas the *APE* maintains high confidence, particularly for pedestrians and cyclists. This confirms the rationale for assigning far-field objects to the *APE*. By leveraging the *MoE Switch* to integrate both experts, the EMC2 system ensures detection reliability across all distances while significantly reducing average inference time and minimizing confidence score variance.



(a) Car Detection Confidence Scores vs. Distance



(b) Pedestrian Detection Confidence Scores vs. Distance



(c) Cyclist Detection Confidence Scores vs. Distance

Figure 9. Detection Confidence Scores vs. Object Distance. Left to right in each subfigure: (1) *Latency-Prioritized Expert*, (2) *Accuracy-Prioritized Expert*, and (3) the EMC2 system, on the validation set. Three well-trained experts keep the confidence high with minimal variance across distances.

5 Limitations

Our current work primarily focuses on optimizing the algorithmic framework and deployment module. To achieve a more comprehensive solution for the algorithm, software, and hardware co-optimization work, we will explore topics related to dataset augmentation and hardware management. These advancements are expected to further enhance system robustness, efficiency, and adaptability in real-world applications.

In terms of dataset expansion, we plan to leverage large language models (LLMs) [97] and other generative AI techniques [98] to synthesize additional edge-case data. Such data augmentation will enable the model to better generalize across diverse and challenging scenarios, reducing algorithmic constraints and improving overall performance. By generating synthetic yet realistic training samples, we aim to mitigate data scarcity issues, enhance robustness to out-of-distribution cases, and lower the reliance on costly real-world data collection. Furthermore, we will explore self-supervised learning strategies to maximize the utility of both synthetic and real-world datasets while minimizing manual annotation efforts.

For hardware management, we will incorporate state-of-the-art load-balancing techniques, such as [99] and [100], to

dynamically allocate computational resources across different threads at varying time intervals. This dynamic resource scheduling will optimize hardware utilization, minimize latency, and ensure efficient workload distribution. This technique will investigate adaptive resource management strategies that leverage real-time performance monitoring and predictive modeling to allocate computing power more effectively. These strategies will allow the system to respond dynamically to variations in computational demand, thereby achieving optimal trade-offs between energy consumption and processing speed.

Moreover, we intend to integrate these optimizations into a scalable and modular architecture, facilitating seamless deployment across diverse hardware platforms. By further bridging the gap between software and hardware optimization, we aspire to create a more efficient, adaptive, and high-performance system capable of addressing the evolving demands of real-world applications.

6 Conclusion

We presented EMC2, a novel multimodal framework designed to achieve a remarkable balance between accuracy and efficiency in 3D object detection for ADS. By incorporating an MoE routing mechnism, Hierarchical Training strategy, and Multiscale Pooling, EMC2 adapts dynamically to varying traffic scenarios, maintaining efficiency without compromising accuracy. The framework demonstrates SOTA performance on the KITTI dataset, particularly excelling in hard-level scenarios, with accuracy improvements of 1.42%, 3.28%, and 6.03% for cars, pedestrians, and bicycles, respectively. Optimized for edge deployment, EMC2 achieves 159.06% faster inference on the Jetson platform, meeting real-time requirements while maintaining high accuracy.

Future work includes the development of a more precise and adaptive MoE switch to enhance decision-making across diverse scenarios. Further optimization of the framework's deployment on edge devices will aim to reduce latency while preserving its performance, ensuring its applicability to real-time autonomous driving systems.

References

1. Mao, H., Yang, X. & Dally, B. A Delay Metric for Video Object Detection: What Average Precision Fails to Tell . In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, 573–582, DOI: 10.1109/ICCV.2019.00066 (IEEE Computer Society, Los Alamitos, CA, USA, 2019).

2. Lin, S.-C. *et al.* The architectural implications of autonomous driving: Constraints and acceleration. In *Proceedings of the Twenty-Third International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS '18, 751–766, DOI: 10.1145/3173162.3173191 (Association for Computing Machinery, New York, NY, USA, 2018).

3. Deng, J. *et al.* Voxel r-cnn: Towards high performance voxel-based 3d object detection. *arXiv:2012.15712* (2020).

4. Zhang, C. *et al.* Optimizing fpga-based accelerator design for deep convolutional neural networks. In *Proceedings of the 2015 ACM/SIGDA international symposium on field-programmable gate arrays*, 161–170 (2015).

5. Chen, T. *et al.* {TVM}: An automated {End-to-End} optimizing compiler for deep learning. In *13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18)*, 578–594 (2018).

6. Cai, J., Wang, Y. & Hwang, J.-N. Ace: Ally complementary experts for solving long-tailed recognition in one-shot. In *Proceedings of the IEEE/CVF international conference on computer vision*, 112–121 (2021).

7. Cai, W. *et al.* A survey on mixture of experts. *arXiv preprint arXiv:2407.06204* (2024).

8. Song, Z. *et al.* Robustness-aware 3d object detection in autonomous driving: A review and outlook. *IEEE Transactions on Intell. Transp. Syst.* 1–30, DOI: 10.1109/TITS.2024.3439557 (2024).

9. Chabot, F., Chaouch, M., Rabarisoa, J., Teulière, C. & Chateau, T. Deep manta: A coarse-to-fine many-task network for joint 2d and 3d vehicle analysis from monocular image. In *CVPR*, 1827–1836, DOI: 10.1109/CVPR.2017.198 (2017).

10. Xiang, Y., Choi, W., Lin, Y. & Savarese, S. Data-driven 3d voxel patterns for object category recognition. In *CVPR*, 1903–1911, DOI: 10.1109/CVPR.2015.7298800 (2015).

11. Li, P., Zhao, H., Liu, P. & Cao, F. Rtm3d: Real-time monocular 3d detection from object keypoints for autonomous driving (2020). arXiv:2001.03343.

12. Min, Z. *et al.* Neurocs: Neural nocs supervision for monocular 3d object localization. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 21404–21414, DOI: 10.1109/CVPR52729.2023.02050 (2023).

13. Zhang, R. *et al.* Monodetr: Depth-guided transformer for monocular 3d object detection. *ICCV 2023* (2022).

14. Brazil, G. *et al.* Omni3D: A large benchmark and model for 3D object detection in the wild. In *CVPR* (IEEE, Vancouver, Canada, 2023).

15. Peng, L., Wu, X., Yang, Z., Liu, H. & Cai, D. Did-m3d: Decoupling instance depth for monocular 3d object detection. In *European Conference on Computer Vision* (2022).

16. Zhang, W., Liu, D., Ma, C. & Cai, W. Alleviating foreground sparsity for semi-supervised monocular 3d object detection. In *2024 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, 7527–7537, DOI: 10.1109/WACV57701.2024.00737 (2024).

17. Wu, Z. *et al.* Fd3d: Exploiting foreground depth map for feature-supervised monocular 3d object detection. *Proc. AAAI Conf. on Artif. Intell.* **38**, 6189–6197, DOI: 10.1609/aaai.v38i6.28436 (2024).

18. Wang, L. *et al.* Depth-conditioned dynamic message propagation for monocular 3d object detection. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 454–463, DOI: 10.1109/CVPR46437.2021.00052 (2021).

19. Ma, X. *et al.* Rethinking pseudo-lidar representation. In *Computer Vision – ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XIII*, 311–327, DOI: 10.1007/978-3-030-58601-0_19 (Springer-Verlag, Berlin, Heidelberg, 2020).

20. Chen, L. *et al.* Shape prior guided instance disparity estimation for 3d object detection. *IEEE Transactions on Pattern Analysis Mach. Intell.* **44**, 5529–5540 (2021).

21. Peng, W., Pan, H., Liu, H. & Sun, Y. Ida-3d: Instance-depth-aware 3d object detection from stereo vision for autonomous driving. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 13012–13021, DOI: 10.1109/CVPR42600.2020.01303 (2020).

22. Shen, Z. *et al.* Digging into uncertainty-based pseudo-label for robust stereo matching. *IEEE Transactions on Pattern Analysis Mach. Intell.* 1–18, DOI: 10.1109/TPAMI.2023.3300976 (2023).

23. Xu, G., Wang, X., Ding, X. & Yang, X. Iterative geometry encoding volume for stereo matching. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 21919–21928 (2023).

24. Xu, G., Wang, Y., Cheng, J., Tang, J. & Yang, X. Accurate and efficient stereo matching via attention concatenation volume. *IEEE Transactions on Pattern Analysis Mach. Intell.* (2023).

25. Chen, Y., Liu, S., Shen, X. & Jia, J. Dsgn: Deep stereo geometry network for 3d object detection. *Proc. IEEE Conf. on Comput. Vis. Pattern Recognit.* (2020).

26. Cheng, X. *et al.* Hierarchical neural architecture search for deep stereo matching. *Adv. Neural Inf. Process. Syst.* **33** (2020).

27. Li, J. *et al.* Practical stereo matching via cascaded recurrent network with adaptive correlation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 16263–16272 (2022).

28. Chen, Q., Sun, L., Cheung, E. & Yuille, A. Every view counts: cross-view consistency in 3d object detection with hybrid-cylindrical-spherical voxelization. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20 (Curran Associates Inc., Red Hook, NY, USA, 2020).

29. Qi, C. R., Chen, X., Litany, O. & Guibas, L. J. Imvotenet: Boosting 3d object detection in point clouds with image votes. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 4403–4412, DOI: 10.1109/CVPR42600.2020.00446 (2020).

30. Qi, C. R., Litany, O., He, K. & Guibas, L. Deep hough voting for 3d object detection in point clouds. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, 9276–9285, DOI: 10.1109/ICCV.2019.00937 (2019).

31. Shi, S., Wang, Z., Shi, J., Wang, X. & Li, H. From points to parts: 3d object detection from point cloud with part-aware and part-aggregation network. *arXiv preprint arXiv:1907.03670* (2019).

32. Lang, A. H. *et al.* Pointpillars: Fast encoders for object detection from point clouds. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 12689–12697, DOI: 10.1109/CVPR.2019.01298 (2019).

33. Shi, S., Wang, X. & Li, H. Pointrcnn: 3d object proposal generation and detection from point cloud. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2019).

34. Li, B. 3d fully convolutional network for vehicle detection in point cloud. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 1513–1518, DOI: 10.1109/IROS.2017.8205955 (2017).

35. Chen, Y., Liu, S., Shen, X. & Jia, J. Fast point r-cnn. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, 9774–9783 (2019).

36. Yan, Y., Mao, Y. & Li, B. Second: Sparsely embedded convolutional detection. *Sensors* **18**, DOI: 10.3390/s18103337 (2018).

37. Zhou, Y. & Tuzel, O. Voxelnet: End-to-end learning for point cloud based 3d object detection. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 4490–4499, DOI: 10.1109/CVPR.2018.00472 (2018).

38. Shi, S. *et al.* Pv-rcnn: Point-voxel feature set abstraction for 3d object detection. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 10526–10535, DOI: 10.1109/CVPR42600.2020.01054 (2020).

39. Yang, Z., Sun, Y., Liu, S. & Jia, J. 3dssd: Point-based 3d single stage object detector. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 11037–11045, DOI: 10.1109/CVPR42600.2020.01105 (2020).

40. Yang, Z., Sun, Y., Liu, S., Shen, X. & Jia, J. Std: Sparse-to-dense 3d object detector for point cloud. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, 1951–1960, DOI: 10.1109/ICCV.2019.00204 (2019).

41. Wang, Y. *et al.* Pillar-based object detection for autonomous driving. In *Computer Vision – ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXII*, 18–34, DOI: 10.1007/978-3-030-58542-6_2 (Springer-Verlag, Berlin, Heidelberg, 2020).

42. Yin, T., Zhou, X. & Krähenbühl, P. Center-based 3d object detection and tracking. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 11779–11788, DOI: 10.1109/CVPR46437.2021.01161 (2021).

43. Huang, T., Liu, Z., Chen, X. & Bai, X. Epnet: Enhancing point features with image semantics for 3d object detection. In Vedaldi, A., Bischof, H., Brox, T. & Frahm, J.-M. (eds.) *Computer Vision – ECCV 2020*, 35–52 (Springer International Publishing, Cham, 2020).

44. Vora, S., Lang, A. H., Helou, B. & Beijbom, O. Pointpainting: Sequential fusion for 3d object detection. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 4603–4611, DOI: 10.1109/CVPR42600.2020.00466 (2020).

45. Nabati, R. & Qi, H. Centerfusion: Center-based radar and camera fusion for 3d object detection. In *2021 IEEE Winter Conference on Applications of Computer Vision (WACV)*, 1526–1535, DOI: 10.1109/WACV48630.2021.00157 (2021).

46. Feng, C. *et al.* Hpv-rcnn: Hybrid point–voxel two-stage network for lidar-based 3-d object detection. *IEEE Transactions on Comput. Soc. Syst.* **10**, 3066–3076, DOI: 10.1109/TCSS.2023.3286543 (2023).

47. Wang, Z., Zhan, W. & Tomizuka, M. Fusing bird's eye view lidar point cloud and front view camera image for 3d object detection. In *2018 IEEE Intelligent Vehicles Symposium (IV)*, 1–6, DOI: 10.1109/IVS.2018.8500387 (2018).

48. Xu, D., Anguelov, D. & Jain, A. Pointfusion: Deep sensor fusion for 3d bounding box estimation. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 244–253, DOI: 10.1109/CVPR.2018.00033 (2018).

49. Chen, Y., Li, Y., Zhang, X., Sun, J. & Jia, J. Focal sparse convolutional networks for 3d object detection. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 5418–5427, DOI: 10.1109/CVPR52688.2022.00535 (2022).

50. Chen, Y., Liu, J., Zhang, X., Qi, X. & Jia, J. Largekernel3d: Scaling up kernels in 3d sparse cnns. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 13488–13498, DOI: 10.1109/CVPR52729.2023.01296 (2023).

51. Li, Y. *et al.* Deepfusion: Lidar-camera deep fusion for multi-modal 3d object detection. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 17161–17170, DOI: 10.1109/CVPR52688.2022.01667 (2022).

52. Liu, Z. *et al.* Bevfusion: Multi-task multi-sensor fusion with unified bird's-eye view representation. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 2774–2781, DOI: 10.1109/ICRA48891.2023.10160968 (2023).

53. Zhang, C., Wang, H., Chen, L., Li, Y. & Cai, Y. Mixedfusion: An efficient multimodal data fusion framework for 3-d object detection and tracking. *IEEE Transactions on Neural Networks Learn. Syst.* 1–15, DOI: 10.1109/TNNLS.2023.3325527 (2023).

54. Yang, Z. *et al.* Deepinteraction++: Multi-modality interaction for autonomous driving. In *Arxiv* (2024).

55. Li, Y. *et al.* Unifying voxel-based representation with transformer for 3d object detection. In *Advances in Neural Information Processing Systems* (2022).

56. Wu, X. *et al.* Sparse fuse dense: Towards high quality 3d detection with depth completion. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 5408–5417, DOI: 10.1109/CVPR52688.2022.00534 (2022).

57. Yan, J. *et al.* Cross modal transformer: Towards fast and robust 3d object detection. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, 18222–18232, DOI: 10.1109/ICCV51070.2023.01675 (2023).

58. Wu, H., Wen, C., Shi, S., Li, X. & Wang, C. Virtual sparse convolution for multimodal 3d object detection. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 21653–21662, DOI: 10.1109/CVPR52729.2023.02074 (2023).

59. Xie, G., Chen, Z., Gao, M., Hu, M. & Qin, X. Ppf-det: Point-pixel fusion for multi-modal 3d object detection. *IEEE Transactions on Intell. Transp. Syst.* **25**, 5598–5611, DOI: 10.1109/TITS.2023.3347078 (2024).

60. Song, Z. *et al.* Robofusion: Towards robust multi-modal 3d obiect detection via sam. *arXiv preprint arXiv:2401.03907* (2024).

61. Song, Z., Jia, C., Yang, L., Wei, H. & Liu, L. Graphalign++: An accurate feature alignment by graph matching for multi-modal 3d object detection. *IEEE Transactions on Circuits Syst. for Video Technol.* **34**, 2619–2632, DOI: 10.1109/TCSVT.2023.3306361 (2024).

62. Shi, S., Wang, Z., Shi, J., Wang, X. & Li, H. From points to parts: 3d object detection from point cloud with part-aware and part-aggregation network. *IEEE transactions on pattern analysis machine intelligence* **43**, 2647–2664 (2020).

63. NVIDIA Corporation. *NVIDIA TensorRT Developer Guide* (2024). Accessed: 2025-02-13.

64. Amit Sabne. *XLA: Compiling Machine Learning for Peak Performance* (2020).

65. Bai, J., Lu, F., Zhang, K. *et al.* Onnx: Open neural network exchange. *https://onnx.ai* (2019).

66. Gao, A. *et al.* Esgn: Efficient stereo geometry network for fast 3d object detection. *IEEE Transactions on Circuits Syst. for Video Technol.* **34**, 2000–2009, DOI: 10.1109/TCSVT.2022.3202810 (2024).

67. Jacobs, R. A., Jordan, M. I., Nowlan, S. J. & Hinton, G. E. Adaptive mixtures of local experts. *Neural computation* **3**, 79–87 (1991).

68. He, J. *et al.* Fastmoe: A fast mixture-of-expert training system. *arXiv preprint arXiv:2103.13262* (2021).

69. Li, J., Jiang, Y., Zhu, Y., Wang, C. & Xu, H. Accelerating distributed {MoE} training and inference with lina. In *2023 USENIX Annual Technical Conference (USENIX ATC 23)*, 945–959 (2023).

70. Rasley, J., Rajbhandari, S., Ruwase, O. & He, Y. Deepspeed: System optimizations enable training deep learning models with over 100 billion parameters. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 3505–3506 (2020).

71. Fedus, W., Zoph, B. & Shazeer, N. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *J. Mach. Learn. Res.* **23**, 1–39 (2022).

72. Du, N. *et al.* Glam: Efficient scaling of language models with mixture-of-experts. In *International Conference on Machine Learning*, 5547–5569 (PMLR, 2022).

73. Yang, A. *et al.* M6-t: Exploring sparse expert models and beyond. *arXiv preprint arXiv:2105.15082* (2021).

74. Hwang, C. *et al.* Tutel: Adaptive mixture-of-experts at scale. *Proc. Mach. Learn. Syst.* **5**, 269–287 (2023).

75. Shoeybi, M. *et al.* Megatron-lm: Training multi-billion parameter language models using model parallelism. *arXiv preprint arXiv:1909.08053* (2019).

76. Community, O. Onnx: Open neural network exchange. https://github.com/onnx/onnx (2024). Accessed: YYYY-MM-DD.

77. Lattner, C. *et al.* Mlir: A compiler infrastructure for the end of moore's law. *arXiv preprint arXiv:2002.11054* (2020).

78. Zunin, V. Intel openvino toolkit for computer vision: Object detection and semantic segmentation. In *2021 International Russian Automation Conference (RusAutoCon)*, 847–851 (IEEE, 2021).

79. Li, J. *et al.* onednn graph compiler: A hybrid approach for high-performance deep learning compilation. In *2024 IEEE/ACM International Symposium on Code Generation and Optimization (CGO)*, 460–470 (IEEE, 2024).

80. Kingma, D. P. & Dhariwal, P. Glow: Generative flow with invertible 1x1 convolutions. *Adv. neural information processing systems* **31** (2018).

81. Ansel, J. *et al.* Pytorch 2: Faster machine learning through dynamic python bytecode transformation and graph compilation. In *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2*, 929–947 (2024).

82. Graham, B. Sparse 3d convolutional neural networks. *arXiv preprint arXiv:1505.02890* (2015).

83. Jamrozik, H. A. *et al.* Reducing network latency using subpages in a global memory environment. *ACM SIGOPS Oper. Syst. Rev.* **30**, 258–267 (1996).

84. Hromkovič, J. *Communication complexity and parallel computing* (Springer Science & Business Media, 2013).

85. Qin, H., Li, Q., Speiser, J., Kraft, P. & Ousterhout, J. Arachne:{Core-Aware} thread management. In *13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18)*, 145–160 (2018).

86. Deng, C., Sui, Y., Liao, S., Qian, X. & Yuan, B. Gospa: An energy-efficient high-performance globally optimized sparse convolutional neural network accelerator. In *2021 ACM/IEEE 48th Annual International Symposium on Computer Architecture (ISCA)*, 1110–1123 (IEEE, 2021).

87. Shi, S. *et al.* Pv-rcnn: Point-voxel feature set abstraction for 3d object detection. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 10529–10538 (2020).

88. Reymann, C. & Lacroix, S. Improving lidar point cloud classification using intensities and multiple echoes. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 5122–5128 (IEEE, 2015).

89. Liu, Z., Arief, M. & Zhao, D. Where should we place lidars on the autonomous vehicle?-an optimal design approach. In *2019 International Conference on Robotics and Automation (ICRA)*, 2793–2799 (IEEE, 2019).

90. Guan, T. *et al.* M3detr: Multi-representation, multi-scale, mutual-relation 3d object detection with transformers. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, 772–782 (2022).

91. Liao, J. *et al.* Gle-net: A global and local ensemble network for aerial object detection. *Int. J. Comput. Intell. Syst.* **15**, 2 (2022).

92. Hoang, H. A. & Yoo, M. 3onet: 3d detector for occluded object under obstructed conditions. *IEEE Sensors J.* (2023).

93. Li, X. *et al.* Logonet: Towards accurate 3d object detection with local-to-global cross-modal fusion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 17524–17534 (2023).

94. Li, Y. *et al.* Voxel field fusion for 3d object detection. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2022).

95. Zhang, Y., Chen, J. & Huang, D. Cat-det: Contrastively augmented transformer for multi-modal 3d object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 908–917 (2022).

96. Wu, H. *et al.* Transformation-equivariant 3d object detection for autonomous driving. In *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 37, 2795–2802 (2023).

97. Kasneci, E. *et al.* Chatgpt for good? on opportunities and challenges of large language models for education. *Learn. individual differences* **103**, 102274 (2023).

98. Fui-Hoon Nah, F., Zheng, R., Cai, J., Siau, K. & Chen, L. Generative ai and chatgpt: Applications, challenges, and ai-human collaboration (2023).

99. Dai, Y., Xu, D., Maharjan, S. & Zhang, Y. Joint load balancing and offloading in vehicular edge computing and networks. *IEEE Internet Things J.* **6**, 4377–4387 (2018).

100. Zhang, J., Guo, H., Liu, J. & Zhang, Y. Task offloading in vehicular edge computing networks: A load-balancing solution. *IEEE Transactions on Veh. Technol.* **69**, 2092–2104 (2019).