AGILE3D: Adaptive Contention- and Content-Aware 3D Object Detection for Embedded GPUs

Pengcheng Wang
wang4495@purdue.edu
Purdue University
West Lafayette, Indiana, USA

Zhuoming Liu
zliu2346@wisc.edu
University of Wisconsin-Madison
Madison, Wisconsin, USA

Shayok Bagchi
bagchi4@purdue.edu
West Lafayette Jr./Sr. High School
West Lafayette, Indiana, USA

Ran Xu
martin.xuran@gmail.com
NVIDIA Corporation
Santa Clara, California, USA

Saurabh Bagchi
sbagchi@purdue.edu
Purdue University
West Lafayette, Indiana, USA

Yin Li
yin.li@wisc.edu
University of Wisconsin-Madison
Madison, Wisconsin, USA

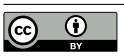
Somali Chaterji
schaterji@purdue.edu
Purdue University
West Lafayette, Indiana, USA

Abstract

Efficient 3D perception is critical for autonomous systems—self-driving vehicles, drones—to navigate safely in dynamic environments. Accurate 3D object detection from LiDAR data must handle irregular, high-volume point clouds, variable latency from contention and scene complexity, and tight embedded GPU constraints. Balancing accuracy and latency under dynamic conditions is crucial, yet existing frameworks like *Chanakya* [NeurIPS '23], *LiteReconfig* [EuroSys '22], and *AdaScale* [MLSys '19] struggle with the unique demands of 3D detection. We present **AGILE3D**, the first adaptive 3D system integrating a cross-model Multi-branch Execution Framework (**MEF**) and a Contention- and Content-Aware Reinforcement Learning-based controller (**CARL**). CARL dynamically selects the optimal execution branch using five novel MEF control knobs: encoding format, spatial resolution, spatial encoding, 3D feature extractor, and detection head. CARL uses supervised training for stable initial policies, then Direct Preference Optimization (DPO) to finetune branch selection without hand-crafted rewards, presenting the first application of DPO to branch scheduling in 3D detection. Comprehensive evaluations show that **AGILE3D** achieves state-of-the-art performance, maintaining high accuracy across varying hardware contention levels and 100-500 ms latency budgets. On NVIDIA Orin and Xavier GPUs, it consistently leads the Pareto frontier, outperforming existing methods for efficient 3D detection.

CCS Concepts

• **Computer systems organization → Embedded software**; • **Computing methodologies → Computer vision problems**; • **Human-centered computing → Ubiquitous and mobile computing systems and tools**.



MobiSys '25, Anaheim, CA, USA

ACM ISBN 979-8-4007-1453-5/2025/06
https://doi.org/10.1145/3711875.3729147

Keywords

3D Object Detection, Embedded GPUs, Point Clouds, Multi-Branch System, Direct Preference Optimization

ACM Reference Format:
Pengcheng Wang, Zhuoming Liu, Shayok Bagchi, Ran Xu, Saurabh Bagchi, Yin Li, and Somali Chaterji. 2025. AGILE3D: Adaptive Contention- and Content-Aware 3D Object Detection for Embedded GPUs. In *The 23rd Annual International Conference on Mobile Systems, Applications and Services (MobiSys '25), June 23–27, 2025, Anaheim, CA, USA.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3711875.3729147

1 Introduction

3D object detection is essential for applications such as autonomous vehicles, delivery drones, robotics, and AR/VR systems, enabling safe navigation and obstacle avoidance [2, 28, 56]. LiDAR technology, which generates 3D point clouds, forms the foundation of these systems. However, processing high-volume, irregular point cloud data on resource-constrained embedded hardware, such as NVIDIA Jetson boards, is challenging [1, 44]. The challenge is exacerbated by dynamically fluctuating resource contention, making it critical to balance accuracy and latency in autonomous systems.

Unlike 2D object detection models that leverage structured image data with stable latency of CNN-based models, 3D detection must contend with the irregularity and sparsity of point clouds, requiring specialized encoders for voxelization and sparse convolutions. These operations significantly increase computational demands, leading to latency variability. For instance, the latest 3D model, DSVT [53], requires 13 TFLOPs of computation per second, far exceeding the NVIDIA Orin GPU's theoretical peak of 5.3 TFLOPs [22], and even more so for less powerful platforms like the NVIDIA Xavier (~1.4 TFLOPs). In practice, real-world deployments rarely achieve peak performance due to resource sharing among concurrent applications, exacerbating latency unpredictability and complicating latency constraints like the 10-20 Hz acquisition rates of modern LiDAR systems [3, 45]. This gap highlights the need for adaptable 3D detection solutions optimized for resource and latency constraints, especially in cost- and energy-sensitive scenarios.

Significant progress has been made in developing 3D object detection models. Early works such as PointNet [32] and PointNet++ [33] pioneered feature extraction from point clouds, while two-stage models like PV-RCNN [42] combined voxelization and point-wise feature abstraction to enhance detection accuracy. Fully trainable models, including VoxelNet [69] and SECOND [60], improved representational capabilities, whereas efficiency-focused designs like PIXOR [61] and PointPillar [23] reduced inference latency. Advanced detection heads like CenterPoint [64] and Part-A^2 [43], alongside transformer-based architectures like DSVT [53], have further pushed detection accuracy. Despite recent progress, existing models are designed and evaluated on server-class GPUs under ideal conditions, neglecting the resource contention and latency constraints of real-world applications. For instance, models such as CenterPoint [64], Part-A^2 [43], and DSVT [53] fail to dynamically adapt to fluctuating resource contention and latency SLOs, falling short of LiDAR operation rates (10-20 Hz), when deployed on a less powerful edge device (*e.g.,* the Nvidia Orin GPU).

In parallel, adaptive systems for 2D workloads have been extensively studied recently, focusing on balancing accuracy and latency under service-level objectives (SLOs) or resource constraints, particularly in video processing tasks. Representative examples include Chanakya [15], LiteReconfig [58], and AdaScale [7], designed for 2D object detection in videos. In 2D detection, branch variations[1] are typically achieved by tuning hyperparameters within a single DNN model, without retraining or structural changes. We term this approach "single model branching," which enables lightweight adjustments ideal for 2D workloads by tuning hyperparameters without retraining. In contrast, "cross-model branching" employs multiple models to address varying requirements.

Extending these 2D techniques to 3D workloads presents two major challenges. **First**, adjustments to parameters like voxel size require retraining of the model, due to the way such changes fundamentally alter the input data representation. For example, variations in voxel or pillar size affect how the point cloud is divided into grids (spatial resolution) and how spatial features are encoded. These shifts disrupt downstream computations, such as sparse convolutions, rendering pre-trained weights incompatible with the modified data structure. Consequently, the model must be retrained or extensively fine-tuned to restore performance, making single-model branching impractical for 3D systems. This limitation necessitates cross-model branching for 3D workloads, enabling dynamic adaptation to diverse input characteristics and resource constraints. Although cross-model branching increases memory usage, the lower memory footprint of 3D models (Sec. 3.1.1) makes it both feasible and advantageous. **Second**, unlike 2D models that process pixels defined on regular grids with stable latency, 3D models handle irregular point clouds, and thus exhibit higher latency variability under resource contention (Sec. 3.1.2). This key difference leads to significant variance in latency when executing the same branch (*i.e.,* the same single model) across different input point clouds even without contention. Such variability necessitates innovative scheduling techniques to dynamically select execution branches in response to changing input content and resource contention.

[1]A branch is a distinct DNN configuration tuned via hyperparameters ("knobs") to ensure consistent latency and accuracy across diverse inputs.

These complexities expose a critical gap: existing frameworks lack the mechanisms to dynamically adapt 3D object detection to simultaneous variations in input content and resource contention. Bridging this gap demands the development of novel, resource-aware systems capable of balancing accuracy and latency at runtime, while adhering to stringent SLOs across diverse deployment environments. Such adaptive 3D detection systems face three key challenges: **First**, embedded devices are resource-constrained and often run multiple applications, leading to resource contention. **Second**, transitioning from 2D to 3D detection requires specialized 3D encoders, which involve operations like voxelization, voxel encoding, and sparse convolutions, significantly increasing system complexity. **Lastly**, systems with tight latency budget must simultaneously handle dynamic external conditions (*e.g.,* content variability across scenes and latency SLOs) and internal constraints (*e.g.,* hardware contention from co-existing applications).

To address these challenges, we present **Agile3D**, *the first adaptive, contention- and content-aware 3D object detection system tailored for embedded GPUs.* At its core, Agile3D employs a Multi-branch Execution Framework (MEF) with five novel control knobs: encoding format, spatial resolution, spatial encoding method, 3D feature extractor variants, and detection heads (Sec. 3.3). These control knobs enable over 50 unique model configurations, allowing the system to adapt its execution strategy based on input data, resource availability, and system SLOs. Notably, the first four of these five control knobs are specifically designed for 3D point cloud object detection, distinguishing Agile3D from previous 2D adaptive frameworks [7, 20, 36, 58, 65]. While the MEF facilitates dynamic operation, the Contention- and Content-Aware RL-based (CARL) controller guarantees system adaptability through fine-grained scheduling. CARL dynamically selects optimal branches at runtime, addressing variability in input content, hardware constraints, and latency SLOs. Traditional RL-based controllers, such as Chanakya [15], depend on human-designed reward functions, which often lead to suboptimal results. CARL overcomes this limitation by employing Direct Preference Optimization (DPO) [35], a method that eliminates the need for manual reward tuning by learning directly from preference comparisons. While DPO is widely used in domains like Large language models (LLMs) with human-labeled "good" and "bad" outputs, CARL adapts this concept for 3D detection by leveraging a heuristic beam search oracle to label optimal branches. This approach replaces the need for extensive manual labeling, ensuring efficient training and robust adaptability. As a result, CARL achieves superior accuracy and adaptability in complex 3D tasks, even under dynamic runtime conditions.

We evaluate Agile3D on three major benchmarks—Waymo [45], nuScenes [3], and KITTI [14]—covering diverse scenarios and complexities, using NVIDIA Jetson Orin and Xavier GPUs. Agile3D consistently achieves high accuracy across latency SLOs (100-500 ms) and contention levels. It outperforms adaptive system controllers such as Chanakya [15] and LiteReconfig [58], and static 3D models like DSVT [53], CenterPoint [64], and PointPillars [23] by 1-5% accuracy, while adhering to the latency constraints.

We summarize our contributions as follows:

1. We present Agile3D, the first adaptive 3D object detection system for embedded GPUs, designed to adapt seamlessly to varying

contention levels and latency SLOs while maintaining robust performance across diverse datasets (*e.g.,* Waymo, nuScenes, and KITTI). AGILE3D leverages five novel control knobs to dynamically optimize latency-accuracy trade-offs, effectively addressing the unique challenges of 3D object detection.

2. We design and implement two controller variants: a CARL controller for dynamic, high-contention environments and a lightweight Look-up Table-based controller for contention-free scenarios. The CARL controller combines supervised training with DPO fine-tuning, eliminating manually tuned rewards and improving accuracy. Using a heuristic beam search, our fine-tuning automatically labels optimal branches, significantly reducing manual effort.

3. AGILE3D delivers significant accuracy gains (1-5%) over state-of-the-art (SOTA) baselines such as *Chanakya* [NeurIPS '23] and *LiteReconfig* [EuroSys '22], while maintaining practical SLOs (100-500 ms) across varying contention levels. It consistently excels on diverse datasets and operates efficiently on NVIDIA Orin and Xavier platforms.

2 Background

2.1 3D Object Detection Algorithms

Point Cloud Data. LiDAR generates unordered, irregular, and sparse point clouds for spatial mapping [24]. Grid-based methods [2] structure this data through voxelization [69], balancing efficiency and computational cost. Hard Voxelization (HV) restricts points per grid cell, while maintaining fixed grid dimensions, causing detail loss in dense areas. Dynamic Voxelization (DV) removes point-per cell caps (allowing unlimited points per cell) but retains fixed grid dimensions. This results in two inefficiencies: dense regions may retain noise rather than discriminative features; sparse regions waste computation on empty grid cells.

Local Processing for Sensors. Efficient sensor data processing on embedded GPUs, like LiDAR and cameras, relies on lightweight DNNs [18, 19, 25, 47, 55, 67]. While resource-efficient, these models lack adaptability to dynamic latency SLOs and input variability, limiting their real-world utility.

2.2 Adaptive 2D Vision Systems

Recent advances in adaptive computer vision systems have focused on addressing dynamic latency SLOs and adapting to varying levels of resource contention by responding intelligently to input content characteristics [7, 15, 20, 21, 36, 59, 65]. Configurations are implemented through dynamic adjustments within a single model [7, 21, 57] or ensembles leveraging multiple models or exits [11, 51]. Customized lightweight networks tailored to datasets further enhance efficiency [10]. These methods balance latency and accuracy, ensuring stable performance under dynamic conditions. While effective for 2D tasks, these approaches face significant challenges when extended to 3D systems. As discussed in Sec. 3.1, transitioning to 3D introduces greater computational demands, irregular data structures, and the need for more sophisticated content reasoning. Unlike 2D systems, where model parameters can be dynamically adjusted without retraining, 3D detection often requires retraining or structural modifications for changes like voxel size. These modifications alter data representations and computation flows, rendering single-model branching infeasible. Furthermore, 3D models experience higher latency variability under resource contention, complicating stable performance in dynamic environments. *These limitations emphasize the need for novel adaptive mechanisms specifically designed for 3D detection. Such mechanisms must account for retraining requirements, fluctuating contention, input variability, and latency constraints to achieve robust performance.*

[2]Here, "grid" refers to both voxel and pillar formats.

2.3 RL with Feedback

Reinforcement Learning with Human Feedback (RLHF) [8, 27, 31] integrates human preferences into LLM training. First, LLMs are pre-trained on large datasets via unsupervised learning. They are then fine-tuned using human-labeled data through supervised learning, followed by further fine-tuning via reinforcement learning that leverages human feedback. A reward model evaluates outputs, and RL techniques like Proximal Policy Optimization (PPO) [41] refine the policy to align with human expectations. Though effective, RLHF requires a complex pipeline with an auxiliary reward model. DPO [35] simplifies this by bypassing the reward model, directly optimizing policy using preference pairs. The model is trained to favor preferred outputs over less favorable ones via a contrastive loss function, offering an efficient mechanism for preference-based optimization. Inspired by RLHF and DPO, we adapt these methods for multi-branch scheduling in AGILE3D. Here, the controller (analogous to the language model) selects optimal branches under latency and contention levels. Using preference pairs generated by an Oracle controller, DPO optimizes branch selection policies without an intermediate reward model, improving system performance by tailoring LLM-inspired techniques to 3D detection.

3 Motivation and Design

Our work addresses the unique challenges of designing an adaptive 3D object detection system. We demonstrate four motivation studies and present the key challenges in Sec. 3.1, provide an overview of AGILE3D in Sec. 3.2, introduce the cross-model MEF in Sec. 3.3, and present our contention- and content-aware RL-based controller design in Sec. 3.4.

3.1 Motivational Studies

3.1.1 2D vs. 3D Detection. To highlight the need for adaptive 3D systems, we compare structural and latency distribution differences between 2D and 3D detectors. While 2D models process dense, structured images, 3D detectors handle unordered, sparse point clouds, requiring a 3D Encoder for spatial feature extraction, which introduces unique latency and computational demands. **Study Setup.** We benchmark widely used 2D models (Faster RCNN [38], Sparse RCNN [46], Dynamic RCNN [66], SSD [26], YOLOF [5], TOOD [12]) and 3D models (SECOND [60], PointPillars [23], CenterPoint [64] with CP-Voxel and CP-Pillar variants). **Results and Findings.** As shown in Fig. 1, latency distributions differ significantly between 2D and 3D models. In 2D, the Backbone dominates latency (47%-78%), followed by the Neck (4%-21%) and Detection Head (16%-47%). For 3D models, the 3D Encoder accounts for 21%-44% of latency, surpassing the Backbone (15%-36%) in absolute computational demand. This highlights the inefficiency of adaptive 2D techniques

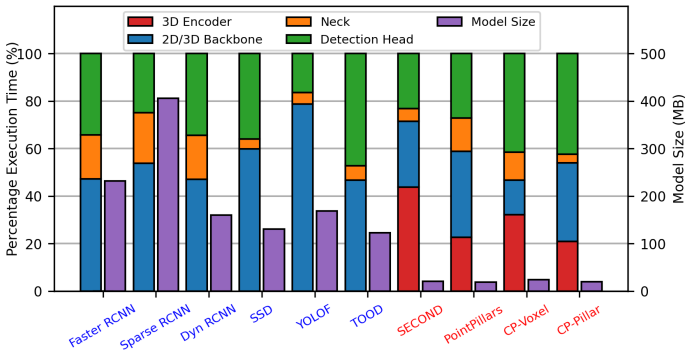


Figure 1: Comparison of execution time and model size for 2D and 3D models. 3D models require higher computation for point clouds but offer better memory efficiency, averaging 20.53 MB versus 203.32 MB for 2D models.

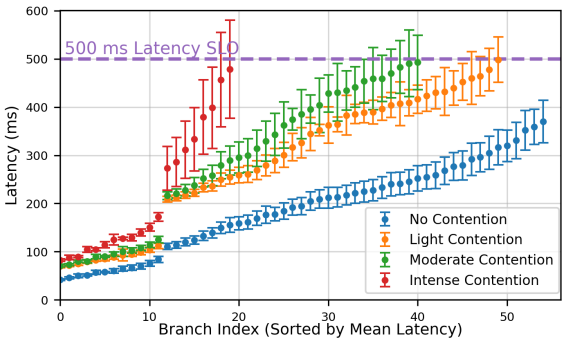


Figure 2: Mean latency with standard deviation across branches. Higher contention increases variability and limits branches within the 500 ms SLO, emphasizing the need for contention- and content-aware 3D controls.

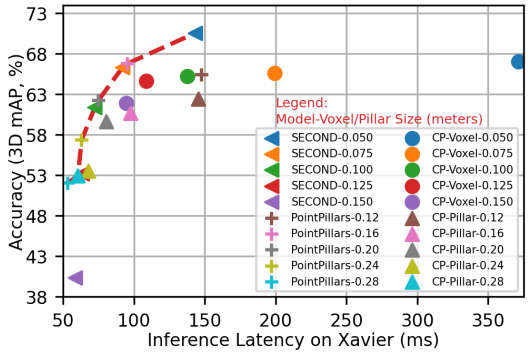


Figure 3: Comparison of 3D models—SECOND, PointPillars, CP-Voxel, and CP-Pillar—at different spatial resolutions. *Key insight: No single model dominates across all latency ranges, motivating the need for adaptive switching among models.*

when applied to 3D systems, as 3D models require specialized encoders to process point clouds into structured spatial features. A counter-intuitive observation is that 3D models are significantly more memory-efficient than 2D models (purple bars in Fig. 1). 2D models on COCO average 203.32 MB, whereas 3D models on KITTI average 20.53 MB—nearly one-tenth the size, despite KITTI point clouds containing 45% of COCO's image data volume. This compactness reflects the efficiency of 3D models in leveraging sparse point clouds and voxelization to capture essential spatial information with fewer parameters. Unlike 2D models, which process dense color, texture, and background information, 3D models focus on spatial structure, efficiently encoding occupied regions and surface geometry, thereby reducing memory requirements.

3.1.2 High Latency Variance of 3D Models. Maintaining low latency violations is crucial in autonomous systems to ensure timely responses across different scenarios. Significant latency variability in 3D models highlights the need for a contention- and content-aware controller. **Study Setup.** We measure the latency of all branches[3] in the MEF on an embedded GPU under different contention levels. Contention levels are calibrated as detailed in Sec. 4.4; higher levels indicate greater resource contention. For each branch, we compute the mean latency, standard deviation, and coefficient of variation (*i.e., stddev/mean*) to capture stability, with lower values indicating greater consistency. Results are reported for branches with mean latencies under 500 ms. **Results and Findings.** As shown in Fig. 2, latency variability increases significantly with higher contention levels. The coefficient of variation ranges from 2.62% to 11.91% under no contention, 2.83% to 13.34% under light contention, 1.94% to 14.13% under moderate contention, and 2.13% to 21.38% under intense contention, with heavier models exhibiting higher variance. Two types of latency variance are observed: 1. *Within-Branch Variance*: Variability caused by differences in input point cloud density. Dense or cluttered point clouds require more processing, increasing latency compared to sparser inputs, worsening as contention increases. 2. *Between-Branch Variance*: Differences arise due to architectural variations across branches. Operations like grouping, sampling, voxel encoding, and sparse/dense convolution introduce varying computational demands, causing model latency variability. 3D models exhibit significant latency variability under resource contention and dynamic inputs. A contention- and content-aware controller is critical to adapt execution paths, reduce latency violations, and ensure reliable performance.

3.1.3 Need for Multi-Model Design. In autonomous systems, latency and accuracy requirements vary based on environmental conditions, system speed, and operational demands. AGILE3D is designed to adapt to these dynamic scenarios, ensuring consistent performance across conditions. **Study Setup.** We evaluate four 3D models: SECOND, PointPillars, CP-Voxel, and CP-Pillar, each tested with five grid sizes on Xavier boards. **Results and Findings.** The observed behaviors, shown in Fig. 3, are as follows: [SECOND]: accuracy 40%-70%, latency 58-143 ms; [PointPillars]: accuracy 52%-65%, latency 53-147 ms; [CP-Voxel]: accuracy 62%-67%, latency 94-371 ms; [CP-Pillar]: accuracy 53%-63%, latency 74-186 ms. The relationship between spatial resolution (grid sizes) and detection accuracy is nuanced, especially for different object sizes and classes. For smaller objects, such as pedestrians, higher spatial resolution (smaller grid sizes) improves accuracy due to better feature representation. For instance, the PointPillars model PP-0.12 achieves higher accuracy in pedestrian detection (44.46%) compared to PP-0.16 (40.24%). However, for larger objects like cyclists and

[3]In AGILE3D, "branch" and "model" are used interchangeably.

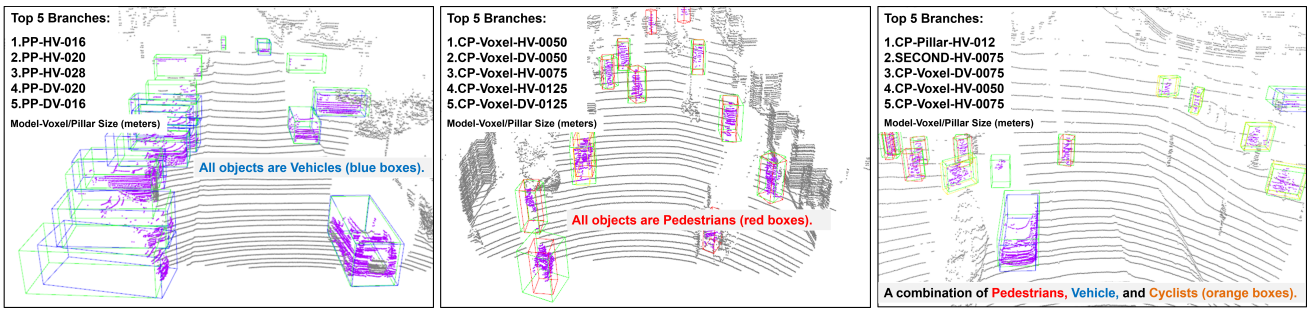


Figure 4: Visualization of diverse point clouds: Vehicles [L], Pedestrians [M], and a mix of Pedestrians, Cyclists, and Vehicles [R]. Ground-truth boxes are green, with top branch predictions for Pedestrians red, Cyclists orange, and Vehicles blue. The top-5 model ranking varies by context: for [L], pillar-based models prevail due to the straightforward geometry; for [M], voxel and center-based detection excel due to their robustness to smaller, varied orientations; and for mixed-object scenes [R], the complexity defies simple explanations, motivating AGILE3D's multi-branch and content-aware controller design.

cars, higher resolution yields fewer gains, as these objects are more easily detectable at lower resolutions due to better global feature aggregation. For example, PP-0.16 outperforms PP-0.12 in detecting cyclists (65.23% vs. 58.73%) and cars (75.98% vs. 69.53%). *No single model consistently occupies the Pareto frontier under all conditions.* This emphasizes the importance of an intelligent system that balances accuracy and latency under varying SLOs to ensure robust performance, and we adopt this approach in designing AGILE3D.

3.1.4 Need for Content-Aware Design. To handle diverse contexts in autonomous systems, we employ a content-aware design to select the best branch at runtime, where each branch is an independent model (Sec. 3.3-3.4). These branches enhance detection accuracy across varied scenarios, highlighting the adaptability of our multi-model and content-aware approach. **Study Setup.** We examine three point clouds with different object compositions: vehicles only (Fig. 4[L]), pedestrians only ([M]), and a mix of pedestrians, cyclists, and vehicles ([R]). Each subfigure lists the top-5 branches by accuracy. **Results and Findings.** Fig. 4 reveals notable variability in the optimal branches across contexts. In vehicle-only scenes ([L]), pillar-based models perform best, as they are suited for simpler environments with large objects and limited vertical detail. In pedestrian-only scenes ([M]), CP-Voxel models excel due to their ability to detect smaller objects with complex vertical features and diverse rotations. In mixed-object contexts ([R]), top-performing models include CP-Pillar, SECOND, and CP-Voxel, highlighting the challenge of selecting a fixed model for complex content. Models with anchor-based Detection Heads work well for axis-aligned objects ([L]), while center-based Detection Heads are better suited for non-axis-aligned objects ([M]). *The variability in optimal model selection across contexts necessitates a flexible, multi-model, and content-aware approach. AGILE3D provides the adaptability to account for the dynamic content in autonomous systems.*

3.1.5 Key Challenges. **Building on the motivational studies, we identify the following challenges:**

• *Techniques from adaptive 2D systems are inadequate for 3D*: Existing adaptive 2D techniques, such as Chanakya [15] and LiteReconfig [58], struggle with 3D due to the differences in model structure and latency distribution. Specialized 3D models require a dedicated 3D Encoder to convert raw data into spatial features, adding complexity absent in 2D tasks. Optimizing 3D system's latency and accuracy performance demands novel approximate mechanisms and a tailored controller design.

• *Inflexibility of the 3D models*: The 3D Encoder plays a critical role in encoding sparse point cloud data into structured spatial features, unlike the 2D images processed directly by CNN-based 2D Backbones. Adjusting voxel or pillar sizes in 3D significantly impacts feature map dimensions and necessitates model retraining due to the fixed input requirements of fully connected layers widely used in the 3D Encoder. Unlike the resizing flexibility of 2D models, this adds rigidity to 3D models, complicating system-level design and requiring specialized handling.

• *Interdependencies in system design*: In AGILE3D, a dependency-driven approach is necessary due to the tight interconnections between the 3D Encoder, 3D Backbone, and Detection Head components. Unlike in 2D systems, where control knobs can be tuned independently, 3D models require coordinated adjustments across all modules. These interdependencies make it essential to design adaptive 3D systems that can dynamically reconfigure the entire pipeline—spanning from the 3D Encoder to the Detection Head—to ensure efficient operation under varying spatial resolutions.

• *Necessity for contention- and content-aware design*: The significant latency variance observed in 3D models, caused by factors like input variability and resource contention, emphasizes the need for dynamic control mechanisms that adapt to changing conditions. Furthermore, the nuanced relationship between accuracy and latency, influenced by input content and model architecture, demonstrates that no single model consistently performs optimally across all scenarios. A robust contention- and content-aware approach is essential, as the optimal model choice varies depending on the context.

3.2 Approach Overview

We design AGILE3D to dynamically adapt to resource contention and input content while meeting strict latency requirements. At its core, AGILE3D features an MEF comprising diverse execution branches managed by the runtime CARL controller. Each branch leverages five tunable modules ("knobs") across critical 3D detection components. These knobs allow AGILE3D to flexibly balance latency and accuracy, extending grid-based 3D detection methods for diverse performance tuning. The system buffers and preheats the MEF in memory on embedded devices during the initial phase, enabling

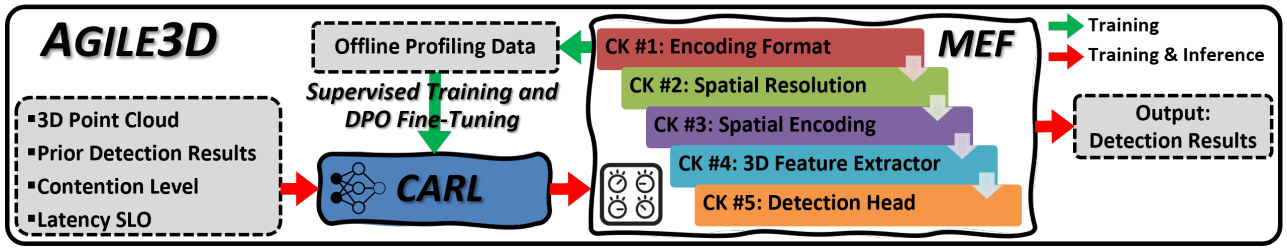


Figure 5: AGILE3D integrates MEF and CARL for dynamic branch selection based on input content, contention levels, and latency SLOs. Supervised training with DPO fine-tuning and five control knobs (CK) ensure adaptability across diverse scenarios.

sub-1 ms branch switching and ensuring timely responsiveness. With the CARL controller's contention- and content-aware strategy, AGILE3D achieves high accuracy and low latency violation ratio across diverse scenarios. Fig. 5 illustrates how MEF and CARL enable AGILE3D to function as the first adaptive 3D detection system operable on embedded GPUs.

3.3 Design of MEF

3.3.1 AGILE3D's Control Knobs. In designing AGILE3D, we carefully select five control knobs based on domain knowledge and our analysis of key stages in the 3D object detection pipeline in Sec. 3.1 (*e.g.,* Fig. 1 highlights the importance of the 3D Encoder, Fig. 3 shows the effects of voxel and pillar sizes, and Fig. 4 demonstrates how different 3D models excel under various scenarios). Compared to traditional knobs for 2D image data [7, 15, 58], these control knobs are tailored to key stages in processing point cloud data, providing a more significant impact on the latency and accuracy of 3D detection models.

#1. **Point Cloud Encoding Format**: Defines how raw point cloud data is encoded, either into voxels (3D cuboids that capture volumetric information) or pillars (vertical columns with no vertical segmentation). Voxel partitioning captures finer spatial details, enhancing accuracy but increasing computation. Pillar partitioning is more efficient but loses some height information, making it suitable for less complex scenes.

#2. **Spatial Resolution**: Adjusts voxel or pillar sizes to control the granularity of spatial information, balancing the trade-off between speed and detail. Larger partitions reduce detail and computational load, while smaller partitions capture more detail at the cost of higher latency.

#3. **Spatial Encoding (HV vs. DV)**: Determines how point clouds are voxelized. HV uses fixed grids, limiting points per grid and total number of grids, and improving stability. DV adapts to data density by eliminating these two limitations, which makes it more dynamic but may sacrifice some stability.

#4. **3D Feature Extractor**: Chooses the neural network type for high dimension 3D feature extraction. Transformers work with both voxel and pillar data for high accuracy but are computationally intensive. Sparse CNNs are effective for voxel-based data, while 2D CNNs suit pillar-based formats, though they lose some 3D detail.

#5. **Detection Head**: Defines the method for object localization and recognition. Anchor-based one uses predefined anchors for efficiency but struggles with diverse object orientations. Center-based one better handles rotated or hybrid objects (*e.g.,* vehicles at intersections), though more computationally demanding.

By incorporating these five knobs, the MEF in AGILE3D offers a highly adaptive and configurable framework, ensuring strong performance for 3D object detection with tight latency budget and under resource contention.

3.3.2 Synergy among Control Knobs. The control knobs are interdependent, impacting both computational efficiency and detection accuracy. The key synergies among these control knobs are:

#1. *Synergy among encoding format, spatial resolution, 3D feature extractor, and detection head*: Choosing a encoding format for point clouds requires compatible resolution, and a 3D feature extractor supporting this format. For instance, if we choose voxels as the encoding format, we need to select an appropriate voxel size from the available options, choose a feature extractor that can efficiently process voxel-based feature maps (*e.g.,* sparse 3D CNN), and adjust the detection head to accommodate the intermediate feature map sizes resulting from different voxel sizes. Such a model may require a simpler Backbone to balance the computational load arising from the increased complexity of 3D Encoder [23, 63, 64]. This constraint helps the system meet target latency and processing speed.

#2. *Spatial resolution and model retraining*: Modifying voxel or pillar sizes changes the dimensions of the model's intermediate feature maps, often requiring retraining of specific feature extraction and prediction layers. 3D Encoders with PointNet-based feature extractors [64, 69] rely on fixed input dimensions in fully connected layers; therefore, adjusting the resolution necessitates separate models retrained from end-to-end to achieve optimal accuracy.

#3. *Impact of spatial encoding on latency and its variability*: The choice between HV and DV impacts the stability of data processing latency in both the spatial encoding step and subsequent modules, including the Backbone and Detection Head. As discussed in Sec. 3.3.1, HV ensures stable latency by using a fixed number of grid cells and points per grid—a strategy that provides predictability at the expense of losing detail in dense regions and incurring computational overhead from processing empty areas. In contrast, DV removes fixed caps, allowing unlimited points per grid cell and dynamically adjusting grid allocation based on input data density. This approach reduces inefficiencies in sparse areas and captures more detail in dense regions. However, DV's reliance on dynamic point aggregation introduces variability in latency, as processing times fluctuate with changes in input density. In summary, HV-based models ensure stable latency but sacrifice accuracy due to fixed grid limits, whereas DV-based models enhance accuracy but compromise latency predictability.

Impact on System Design. These dependencies indicate that tuning one control knob may require adjustments to others, rendering it impractical to optimize these parameters within a single model as is common in 2D systems. AGILE3D addresses this challenge by employing cross-model branching, with each branch optimized for a specific set of control knob configurations. This approach enables broad adaptability to meet diverse operational requirements.

3.4 Design of Controller

The controller's main objective is to dynamically select the optimal branch at each timestamp that satisfies the latency SLO and maximizes accuracy, given the current input point cloud and hardware contention. The optimization can be formulated as:

$$b_{opt} = \text{argmax}_{b \in \mathcal{B}}\ acc(b, X, C) \ \text{ s.t. } \ l(b, X, C) + l_c + l_o \leq l_s, \quad (1)$$

where $\mathcal{B}$ denotes the set of available branches, X represents the input point cloud, C is the contention level, and $acc(b, X, C)$ and $l(b, X, C)$ are the accuracy and latency of branch b, respectively. Here, l_c denotes the controller's latency cost, l_o is the branch-switching overhead, and l_s indicates the latency SLO.

A direct solution to this optimization problem is impractical at runtime due to latency that varies dynamically with input content and contention, as well as unknown accuracy beforehand. Therefore, we employ RL techniques to predict the optimal branch. This approach involves two phases: offline training and online prediction. In the offline phase, each branch undergoes profiling on embedded GPUs using a previously unseen dataset, enabling the controller to learn input- and branch-specific latency and accuracy characteristics. During the online phase, the trained controller selects the optimal branch that meets the latency SLO and maximizes accuracy. Next, we detail our controller design.

3.4.1 CARL Controller. Our CARL controller dynamically schedules tasks by considering contention levels and input content. It employs supervised training for initial learning, followed by DPO fine-tuning with preference labels provided by the Approximate Oracle controller using Beam Search (AOB, Sec. 3.4.2). DPO refines branch selection through preference comparisons instead of absolute scores, ensuring efficient optimization.

We frame branch scheduling as a Markov Decision Process (MDP), and consider learning CARL using RL. Formally, the states, actions, and rewards for the MDP are defined as follows. At each timestep i ($i = 0, \ldots, t$), the *state* S_i comprises the current input point cloud X_i, previous detection results D_i, and the current contention level C_i. Formally, the state is represented as $S_i = (X_i, D_i, C_i)$. The *actions* are the branch selections from the MEF, defined as $b_i \in \mathcal{B}$ where $\mathcal{B}$ is the set of all possible branches. CARL selects a proper branch b_i based on the current state S_i so as to optimize the overall efficiency and accuracy. Unlike standard RL, we do not assume an explicit reward for each action, as rewards will be implicitly provided by preference-based optimization in DPO. Further, we consider a discrete set of latency SLOs $\{l\}$ and train separate models for individual latency SLOs.

CARL's Structure. The controller comprises a policy model, which is updated during training, and a reference model, which remains frozen and serves as a stable baseline for calculating the DPO loss,

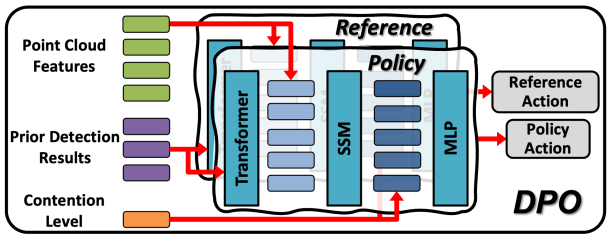


Figure 6: The CARL controller uses a shared architecture for policy and reference models, integrating GD-MAE for 3D features, transformers for prior detection results embedding, SSM for sequence processing, and positional embeddings for latency objectives, enabling adaptive branch selection.

as shown in Fig. 6. Both models share the same architecture, mapping an input state S_i to a probability of actions (*i.e.,* choosing branches) $p(b_i|S_i)$. The model starts with raw point clouds X_i using GD-MAE [62], an efficient 3D feature extraction framework. GD-MAE leverages sparse representations and self-supervised masked autoencoder pre-training on LiDAR data to learn unbiased geometric features. This approach reduces the reliance on labeled data and enhances generalization performance in downstream tasks. The extracted features are combined with tokens generated from Transformer [52] layers, which encode previous detection results D_i into embeddings. These embeddings provide historical context for informed decision-making. The combined features are passed to a Structured State-Space Model (SSM) [16], enabling the controller to model temporal dependencies across consecutive frames. The SSM output is enhanced with a positional embedding representing contention C_i, which encodes the current contention level in the feature space. Finally, the concatenated features pass through a multi-layer perceptron (MLP) to generate the action distribution, enabling effective branch selection aligned with latency objectives.

Supervised Training. We use supervised learning to train the CARL controller, aligning its initial policy with the Oracle's target action b_{opt} (provided by AOB, Sec. 3.4.2) for a given system state S, predicting an action distribution $\pi(b|S)$. The training objective is to minimize the cross-entropy loss between the prediction and the target shown in Eq. 2:

$$\min_{\pi}\ \mathcal{L}_{CE}(\pi(b|S),\ b_{\text{opt}}), \quad (2)$$

DPO Training. While supervised learning optimizes decisions at each timestamp independently, DPO enhances the CARL controller by considering sequences of actions over time. This approach enables the controller to balance short-term and long-term trade-offs, achieving superior overall performance. We initialize the policy model π and reference models π_{ref} from the supervised-trained model. For each state S, we generate positive-negative action pairs (b_p, b_n). The positive action b_p is selected using the AOB, while the negative action b_n is sampled from the reference model π_{ref}. As a result, the positive actions are often more favorable than the negative ones. The policy π is our controller. In traditional DPO training, both positive and negative actions are derived from the reference model, with human annotations used to identify the positive action. In our approach, the AOB replaces human annotations

for determining the positive action, but the underlying preference-comparison mechanism remains unchanged. The training objective is formulated as maximizing the expected log-sigmoid difference between the preferred and non-preferred branch probabilities relative to a reference policy shown in Eq.3.

$$\max_{\pi} \mathbb{E}_{(S,b_p,b_n)\sim\mathcal{D}} \log\sigma\left(\beta\log\frac{\pi(b_p|S)}{\pi_{\text{ref}}(b_p|S)} - \beta\log\frac{\pi(b_n|S)}{\pi_{\text{ref}}(b_n|S)}\right), \quad (3)$$

where β is a hyperparameter controlling the degree of divergence from the reference model π_{ref}, and $\mathcal{D}$ represents the dataset of preference comparisons. This training objective leverages preference comparisons to fine-tune the policy model, aligning it with the AOB controller's decisions and thereby enhancing sequence-level performance. During training, the target policy π is updated, while the reference policy π_{ref} is kept frozen.

3.4.2 Oracle Controller. The Oracle controller represents the theoretical upper bound for content-aware scheduling by selecting the optimal branch for each point cloud with full knowledge of ground-truth accuracy. However, implementing such an Oracle is infeasible, as identifying the optimal branch under contention and latency constraints requires an exhaustive search across all branch combinations—a computationally prohibitive task. To approximate, we employ an AOB, which efficiently identifies near-optimal branch schedules by iteratively refining a limited set of top candidates. AOB serves two main purposes: #1. **Training Labels for CARL:** AOB generates optimal branch selections per frame, used to fine-tune CARL via DPO, helping CARL approximate Oracle-level performance in dynamic environments. #2. **Benchmarking:** Comparing Agile3D's performance to AOB provides insights into adaptability and areas for optimization under varying conditions.

3.4.3 Online Distribution-Aware Look-Up Table (DA-LUT)-based Controller. We also consider a baseline controller with a distribution-aware look-up table to address branch execution variability (Sec. 3.1.2). This DA-LUT controller leverages offline profiling data (mean/-variance of latency and accuracy) for efficient branch selection. Assuming Gaussian latency distributions, it calculates confidence levels (*e.g.,* 99%) to minimize latency violations while maintaining SLOs. The controller stores key-value pairs in the format <*branch, contention, latency mean, latency std, accuracy*> and incurs only 1 ms overhead. Lightweight and content-agnostic, the DA-LUT controller excels in low-contention scenarios where latency fluctuations are minimal, outperforming baselines without requiring complex content reasoning.

4 Implementation

4.1 Hardware and Software

Hardware. We train Agile3D on NVIDIA A100 GPUs and evaluate it on two NVIDIA Jetson platforms: Orin: 12-core ARM CPU, 2048-core Volta GPU, 64GB RAM; Xavier: 8-core ARM CPU, 512-core Volta GPU, 32GB RAM. For stable performance, we set both platforms to max power mode and disable Dynamic Voltage and Frequency Scaling. **Software**: We develop Agile3D using Python and PyTorch, based on the OpenPCDet [50] codebases. Artifacts are open-sourced at https://doi.org/10.5281/zenodo.15073471

Table 1: 2D models' contention levels with concurrent 3D workloads. Transformer-based models (*e.g.*, ViT [9]) exhibit higher contention than convolutional models (*e.g.*, MobileNet [34], EfficientNet [48]).

Models	MobileNet			EfficientNet			ResNet50	ViT	
Variants	V4	V3	V2	B0	B3	B5	N/A	Medium	Base
Contention Levels (%)	32	36	31	28	30	42	43	62	69

4.2 MEF Training

We construct MEF by integrating and enhancing a diverse set of 3D detectors, including DSVT [53], CenterPoint [64], DV [68], PointPillars [23], and SECOND [60]. Rather than naïvely aggregating these models, Agile3D systematically calibrates and optimizes each component to achieve a balanced trade-off between inference latency and detection accuracy. To ensure optimal performance, we calibrate voxel-based detectors along the x and y dimensions (0.1–0.9 m) and adjust the z-dimension (0.1–0.2 m) to align with LiDAR sensor configurations. For pillar-based models, we calibrate the x and y dimensions (0.24–0.9 m) while preserving z-heights according to dataset specifications. This distinction arises because voxel-based models require finer z-granularity to capture vertical details, whereas pillar-based models prioritize lateral coverage. We tune the detectors in MEF to leverage their strengths in addressing the challenges of 3D point clouds. For large datasets like Waymo and nuScenes, we employ DV for efficiency and center-based heads for complex scenes. Each detector uses model-specific setups with standardized preprocessing and augmentations (*e.g.,* rotation, scaling, and flipping along the X and Y axes). Our original MEF consists of nearly 100 branches, which we prune down to approximately 50 based on profiling results on a hold-out profiling set. This process retains only those branches near the Pareto frontier during offline profiling, and reduces overall memory consumption.

4.3 Contention Generator (CG)

We enhance the GPU CGs from Chanakya [15] and LiteReconfig [58], adapting them to simulate real-world workloads better. Building upon the original CGs, our enhancements introduce synthetic contention to mimic the contention level from several 2D models running concurrently with the main 3D task. This setup emulates practical embedded systems where 2D models (*e.g.,* processing camera data) and 3D models (*e.g.,* processing LiDAR data) share GPU resources. Given the diversity of 3D models and their varying sensitivity to contention, we streamline the evaluation by selecting a representative medium-compute-intensity 3D model (DSVT-Pillar with pillar size 0.66). Additionally, prior CG measures GPU utilization offline as a standalone process, neglecting resource sharing during concurrent execution [13]. This approach fails to capture real contention on mobile GPUs accurately. To address this limitation, we introduce a metric that quantifies the latency impact of CG on the primary 3D task, defined as Contention Level = $(1 - L_{\text{wo}}/L_{\text{w}}) * 100\%$, where L_{w} / L_{wo} denote the latency with and without contention, respectively.

Table 1 summarizes the contention levels induced by common 2D models, which range from 28% to 69%, lighter from the mobile CNN models and heavier from the vision transformer models. Given the significant variability in contention levels, we select four representative levels—[38, 45, 64, 67]%—from our CG for evaluation.

To enhance clarity and usability, we categorize these levels as *Light, Moderate, Intense, and Peak*, which are used consistently throughout our experiments.

4.4 CARL Training

We collect latency and accuracy data to train the controller, using distinct datasets to avoid overfitting. Training, profiling, and testing datasets are split by time of day, weather, and location to ensure diverse coverage, with each set containing samples from different conditions that help mitigate overfitting to specific scenarios. We profile all branches on two embedded GPUs under varying contention levels, recording per-sample inference latency. The sample-level accuracy trains the CARL controller to make intelligent decisions, while the data set-level accuracy guides the DA-LUT controller, ensuring efficient decisions for less dynamic scenarios. During CARL controller training, we sample sequences of consecutive point clouds X from the offline profiling data and randomly generate contention levels c_i for each sequence. The controller takes the state as input and selects a branch as the action. We retrieve the corresponding latency and accuracy from the offline profiling data. We train the controller 318,900 episodes using the AdamW optimizer with batch size 16 and a learning rate of 1e-5.

5 Evaluation and Results

We present our experiment setup in Sec. 5.1. In Sec. 5.2, we compare AGILE3D with prior adaptive controllers under various resource contention conditions and latency SLOs. In Sec. 5.3, we evaluate AGILE3D under varying contention levels. In Sec. 5.4, we compare AGILE3D with SOTA static models in contention-free scenarios. In Sec. 5.5, we examine different controller training strategies for AGILE3D. In Sec. 5.6, we demonstrate the effectiveness of our control knobs. In Sec. 5.7, we present several microbenchmark results, including the system overhead, Pareto frontier distributions, and the influence of voxel/pillar size on model performance.

5.1 Experimental Setup

Datasets. We primarily evaluate AGILE3D on Waymo dataset [45]—one of the largest datasets for point cloud based 3D object detection in urban driving scenarios. To demonstrate the generalization of AGILE3D, we also evaluate two other driving datasets: nuScenes [3] and KITTI [14]. *Waymo* includes 1,150 sequences, with 798 / 202 / 150 for training / validation / testing. We split the training set further (637 / 161 for training / profiling) and use the validation set for testing. *nuScenes* contains 1,000 sequences, with 700 / 150 / 150 for training / validation / testing. We partition the training set into 630 / 70 for training / profiling, and use the validation set for testing. *KITTI* has 7,481 training and 7,518 testing samples. The training set is split into 3,340 / 372 / 3,769 for training / profiling / testing. These benchmarks lack annotations for testing sets, thus reporting results on the validation set is a standard practice [6, 53, 58, 64, 69]. Our splits ensure rigorous evaluation with unseen test data.

Metrics. *Waymo*: mean Average Precision (mAP) with IoU thresholds of 0.7 (vehicles) and 0.5 (pedestrians/cyclists) for LEVEL2 difficulty. *nuScenes*: NuScenes Detection Score (NDS) combines mAP with five complementary metrics for comprehensive evaluation. *KITTI*: mAP is averaged across classes and difficulty levels at 40 recall positions.

Adaptive Controller Baselines. Our controller baselines include adaptive methods for 3D object detection, leveraging the same MEF. *Chanakya* [15]: originally designed for 2D workloads, we adapt its RL-based, content-aware control approach to handle the increased complexity and dynamic nature of 3D point cloud processing. This adaptation involves carefully engineering the reward structure and integrating it with MEF, which requires significant effort due to the higher dimensionality and dynamic contention characteristics inherent to the 3D setting. *LiteReconfig* [58]: originally designed as a lightweight contention- and content-aware controller for video detection, LiteReconfig is extended to 3D workloads by recalibrating its contention sensitivity and content-awareness mechanisms. This recalibration is non-trivial, given the higher dimensionality, dynamic contention patterns, and increased computational complexity inherent in 3D tasks. *DA-LUT*: Our LUT based controller described in Sec. 3.4.3. *Oracle (AOB)*: The oracle controller described in Sec. 3.4.2. By using the same set of control knobs for all methods, these controller baselines comprehensively cover a range of scheduling strategies. DA-LUT is based on LUT, LiteReconfig uses supervised learning, and Chanakya considers RL. In addition, AOB provides an upper bound.

Static Model Baselines. We include seven static 3D models as our baselines when comparing with AGILE3D under contention-free scenarios, including *DSVT* [53]: a Transformer-based model with Voxel and Pillar variants, highlighting SOTA 3D encoders; *CenterPoint (CP)* [64]: Voxel and Pillar variants with center-based heads for robust object localization; *Part-A^2* [43]: a two-stage detector refining proposals for better accuracy and box scoring; *PointPillars (PP)* [23]: an efficient 2D convolution-based 3D detector; *SSN* [70]: an extension to PP with shape-aware grouping for improved geometric features; *PV-RCNN* [42]: a combination of voxel and point-based abstraction for enhanced detection accuracy. *SECOND* [60]: a model using sparse convolutions for efficient voxel-based processing.

5.2 Accuracy-Contention Pareto Frontier

Our main experiment evaluates the end-to-end performance of AGILE3D under varying contention levels (*Light, Moderate, Intense, and Peak*) and across multiple latency SLOs (500, 350, and 100 ms) on the Orin GPU using the Waymo dataset. These latency SLOs are designed for driving scenarios, where LiDAR point clouds are typically acquired at 10 Hz [45], whereas many existing systems process these point clouds at only 2 Hz [3]. Moreover, these latency SLOs are challenging for 3D detection on mobile devices even without contention (see Sec. 3.1.2).

The accuracy of AGILE3D, in comparison to the baselines, are summarized in Fig. 7. [CARL+MEF] denotes our full design, while [DA-LUT+MEF] represents a simpler variant. We combine the prior adaptive controllers Chanakya and LiteReconfig with our MEF and retrain them, resulting in [Chanakya+MEF] and [LiteReconfig+MEF]. Across all latency SLOs, AGILE3D maintains a latency violation ratio below 10% and outperforms all adaptive controller baselines by a noticeable margin. For example, under *Intense* contention, AGILE3D achieves 1.6-3% higher accuracy than the best

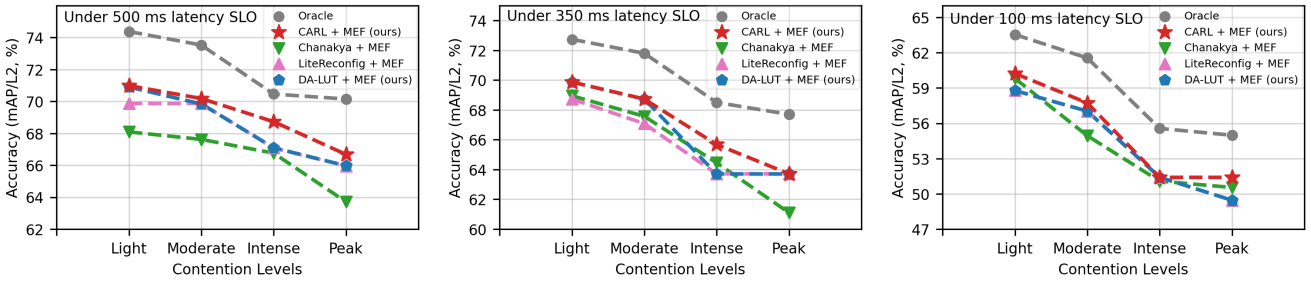


Figure 7: End-to-end evaluation of AGILE3D across varying contention levels (*Light* / *Moderate* / *Intense* / *Peak*) and latency SLOs (500 ms [L], 350 ms [M], and 100 ms [R]) using the Waymo dataset and on Orin GPU. AGILE3D consistently achieves superior accuracy, shining on the Pareto frontier across all contention levels and latency SLOs.

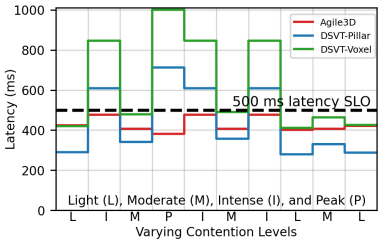


Figure 8: AGILE3D adapts to changing contention levels on the Waymo test set on Orin under 500 ms latency SLO. Baselines fail to adapt to dynamism.

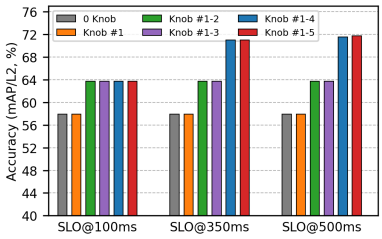


Figure 9: AGILE3D on Waymo (Orin) under three latency SLOs (100, 350, 500 ms): Activating more control knobs improves accuracy and satisfies lower latency SLOs.

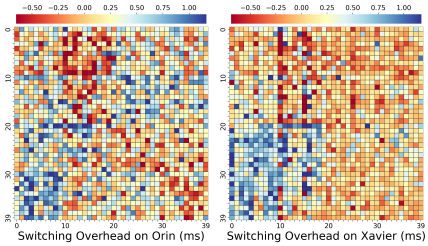


Figure 10: Switching overhead between branches (on Orin, Xavier). Y-axis: source, X-axis: destination branches. Mean overhead <1 ms with pre-buffered models.

adaptive method while meeting latency requirements. It is noteworthy that although AGILE3D underperforms the oracle AOB, the performance gap is limited to 2–5%. Chanakya's original design does not consider hard latency SLOs, thus leading to the worst performance. Collectively, these results highlight AGILE3D's superior performance in a critical real-world application domain—autonomous driving—despite device contention and tight latency SLOs. Under contention scenarios, static approaches fail in most cases; therefore, they have been omitted from Fig. 7 for clarity.

5.3 Adapting to Dynamic Contention

AGILE3D features the ability to adapt to dynamic contention changes on the fly. We further evaluate this ability by simulating dynamic contention levels using the Waymo test set. Specifically, we split the test set into ten segments and process each segment under randomly shuffled contention levels, ensuring compliance with the 500 ms latency SLO. We perform smoothing within each contention level region for ease of interpretation, but observe that even the fluctuations rarely violate the latency SLO. Fig. 8 illustrates that AGILE3D dynamically adjusts to changing contention on the fly, meeting latency requirements while optimizing performance. Static models like DSVT-Pillar and DSVT-Voxel fail to adapt, either violating the latency SLO or under-utilizing the latency budget. These results highlight AGILE3D's strong capability to respond to dynamic conditions changes at runtime.

5.4 Accuracy-Latency Pareto Frontier

We further evaluate the performance of AGILE3D without contention. By removing contention, this scenario offers a theoretically interesting case for assessing accuracy-latency trade-offs and comparing AGILE3D to SOTA static 3D object detection models. Due to the dense LiDAR data, experiments on the Waymo and nuScenes datasets are conducted on the Orin platform. In contrast, the KITTI dataset, with its lower data density and smaller detection range, is evaluated on the more resource-constrained Xavier platform. Additionally, we focus on DA-LUT for nuScenes and KITTI datasets due to their limited annotated data.

Figs. 11 to 13 illustrate AGILE3D's accuracy-latency trade-offs versus baseline 3D models under no contention, evaluated across three datasets. On Waymo (Orin, Fig. 11), AGILE3D exceeds baselines in both accuracy and latency, demonstrating adaptability within the 50 to 350 ms SLO range. Given the 200 ms SLO as an example, while the baselines SECOND, PP, CP-Pillar, and CP-Voxel meet the latency SLO, AGILE3D achieves superior accuracy, surpassing them by 4-11%. For nuScenes (Orin, Fig. 12), AGILE3D outperforms all baselines in both accuracy and inference speed (2-4X faster speed, 7-16% higher accuracy). Additionally, it achieves accuracy levels comparable to DSVT-Pillar while majorly improved speed (1.3x). The same insights can be observed from the KITTI dataset (Xavier, Fig. 13), AGILE3D can always satisfy the 150 ms SLO, while most of the baselines fail. While PP and CP-Pillar variants meet latency SLOs, AGILE3D surpasses them in accuracy by 3-7%. These trends hold consistently across all datasets, highlighting AGILE3D's superiority in dynamic settings.

5.5 Comparing Training Strategies

A key design choice of AGILE3D lies in its training strategy—a combination of supervised pre-training with DPO fine-tuning. This is a sharp contrast to DA-LUT (statistical modeling), LiteReconfig (supervised training) and Chanakya (Q-learning). We fix the control knobs and evaluate the effects of training strategies. Table 2 presents the results on the Waymo dataset and using Orin GPU.

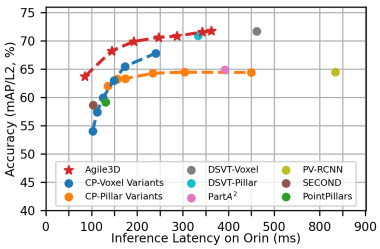


Figure 11: Waymo Performance (Orin). AGILE3D achieves 1-10% higher accuracy than DSVT, CP, PartA^2, SECOND, and PP while adapting to 100-400 ms SLOs – operating 2.8-8X faster than baselines (230-850 ms for the same 64% mAP).

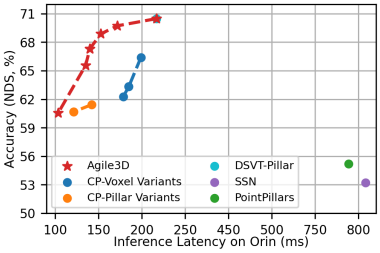


Figure 12: nuScenes Performance (Orin). AGILE3D demonstrates 4-16% accuracy gain over CP, SSN, and PP, while meeting 100-250 ms SLOs, outperforming baselines needing 120-800 ms (1.2-4X slower).

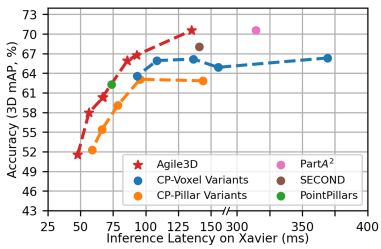


Figure 13: KITTI Performance (Xavier). AGILE3D maintains 2-7% higher accuracy than CP, PP, and SECOND under 50-150 ms latency SLOs, where baselines require 60-375 ms (1.3-2.3X slower).

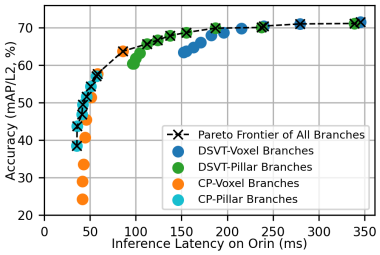


Figure 14: Pareto Frontier: DSVT branches excel in accuracy, while CP branches optimize latency, balancing performance.

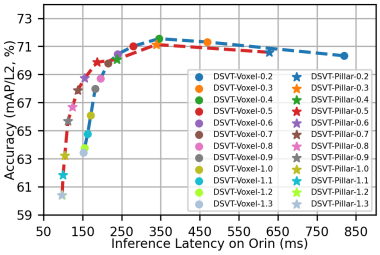


Figure 15: Voxel/pillar size vs. performance: Smaller sizes do *not* consistently improve accuracy despite higher costs.

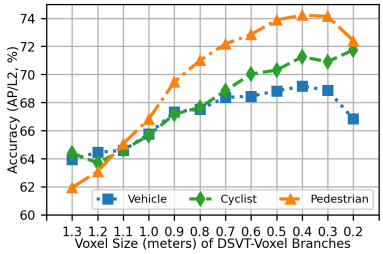


Figure 16: Smaller voxels enhance AP for smaller objects (*e.g.,* pedestrians and cyclists) but offer diminishing returns for vehicles.

We additionally include the vanilla LUT that ignores the variance of latency, as well as the oracle AOB as the upper bound. While vanilla LUT achieves top accuracy in some cases, it suffers from the highest latency violations (up to 49.95%). Supervised learning (LiteReconfig) and RL (Chanakya) both reduce the violation rate, yet at the cost of decreased accuracy. Instead, AGILE3D's training strategy (supervised learning with DPO fine-tuning) strikes a balance, achieving robust performance with low latency violations across varying contention levels.

Table 2: A comparison among training strategies including oracle AOB, vanilla LUT, statistical modeling (stat), supervised learning (SL), reinforcement learning (RL), and supervised learning followed by DPO fine-tuning (SL+DPO). Accuracy (%), latency (ms), and latency violation rate (%) under 500 ms latency SLO are reported using Orin GPU. Gray: infeasible settings with either latency violations over 10% or using an oracle; Bold: best accuracy within the 10% limit.

Controller	Light	Moderate	Intense	Peak
AOB	74.38/385/0.65	73.53/378/0.61	70.46/364/0.09	70.15/359/1.14
Vanilla LUT	71.45/506/48.76	70.90/501/49.95	68.97/505/48.35	68.21/472/37.41
Stat	70.90/430/3.29	69.84/381/0.10	67.10/340/0.63	65.97/328/0.74
SL	69.87/285/0.00	69.87/340/0.00	67.08/340/0.63	65.96/328/0.74
RL	68.09/347/0.34	67.62/262/0.37	66.76/347/0.23	63.71/181/0.00
SL+DPO	**70.99**/415/5.27	**70.18**/407/0.14	**68.73**/477/1.14	**66.68**/362/5.64

5.6 Effects of Control Knobs

Moving forward, we benchmark the effects of control knobs on AGILE3D's performance under varying conditions. Fig. 9 illustrates the accuracy under various latency SLOs (100 ms, 350 ms, and 500 ms) with different control knobs, using the Waymo dataset and Orin platform. The results suggest that activating more control knobs enables AGILE3D to meet stricter latency SLOs and improve accuracy. Higher latency SLOs provide additional slack, further boosting performance with the same number of knobs. We conduct this experiment using both Orin and Xavier, observing a similar trend. However, detailed results are omitted due to space constraints. These findings supports our design of control knobs (Sec. 3.3.1) and demonstrates the role of these knobs in adapting and optimizing performance across datasets and hardware platforms.

5.7 Microbenchmarks

System Overhead. AGILE3D introduce system overhead in three respects: memory to buffer all branches, switching overhead within the MEF, and controller overhead. AGILE3D buffers all MEF branches (*i.e.,* individual models) in memory, because of the efficient 3D model structures discussed in Sec. 3.1.1, the MEF uses <8 GB of RAM, well below the memory capacity of modern mobile devices. The switching between branches will introduce a minor branch-switching overhead. Fig. 10[L], [R] shows this overhead on Orin and Xavier. Pre-buffering limits overhead to under 1 ms, as transitions only require memory to GPU cache operations. In contrast, loading models from disk causes latency spikes exceeding 200 ms. Disk to GPU cache switching costs are 2,394x higher on Xavier (335.16 ms vs. 0.14 ms) and 839x higher on Orin (209.86 ms vs. 0.25 ms). During inference, the controller does not need to be triggered on every point cloud because of the consistency of consecutive point clouds, the average overhead from the controller is about 1ms for DA-LUT or 8ms for CARL. The total overheads represent only a small fraction of the total latency budget (200–500 ms).

Pareto Frontier Distributions. Fig. 14 presents the Pareto frontiers of all branches, reported on Waymo using Orin. The results illustrate individual branch contributions to the accuracy-latency

spectrum. DSVT branches dominate the high-accuracy region, reflecting their precision, while CP branches excel in the low-latency region due to their efficiency. Voxel-based models achieve the highest accuracy, whereas pillar-based models prioritize efficiency.

Voxel/Pillar Size vs. Performance. Figs. 15 and 16 illustrates the impact of voxel/pillar sizes on DSVT performance using the Waymo data on Orin. Smaller voxel sizes theoretically offer higher resolution but do not consistently enhance accuracy. Pedestrians and cyclists are more sensitive to voxel size, with accuracy ranging from 62-74% and 64-72%, respectively, while vehicles show limited variation (64-69%). Overly fine-grained voxelization struggles to capture holistic spatial patterns needed for larger objects. Additionally, smaller sizes increase computational workload and latency, producing larger intermediate feature maps that limit efficiency gains despite theoretical benefits.

6 Related Works

Adaptive Vision Systems for Mobiles. Efficient data processing for LiDAR or cameras on resource-limited mobile devices poses a significant challenge due to strict latency requirements and limited computational resources. Lightweight DNNs, whether handcrafted [18, 19, 67] or designed via neural-architecture search methods [25, 47, 55], address resource limits yet fundamentally lack runtime adaptability to varying SLOs or input content. Recent works introduce adaptability, either within single models [7, 21, 57, 59] or ensembles [11], leveraging techniques like early exits [51, 57], input simplification [29, 37], mixture of experts [39], or task-specific designs [10]. Specifically, adaptive 2D object detection has been explored in video domains [7, 15, 58, 59], often employing multi-branch designs [20, 36, 65]. However, as detailed in Sec. 3.1, these techniques are inadequate for 3D detection due to sparse data structures and irregular computations in point clouds, leading to high variance in latencies.

Systems for Serving DNN Models. The systems community has explored model selection techniques to satisfy latency and accuracy SLOs. INFaaS [40] automates model and hardware selections for cloud platforms like AWS but is unsuitable for embedded devices or the streaming data. Clockwork [17] ensures tail-latency SLOs when scheduling DNNs on GPUs in cloud environments but lacks mobile deployment. Jellyfish [30] combines data and DNN adaptation for latency guarantees in edge networks, relying on desktop-level GPUs. OFA [4] trains a versatile model pruned for deployment but lacks runtime adaptability. HAT [54] optimizes transformers for specific hardware pre-deployment, while ElasticViT [49] uses NAS to train ViT supernets and select optimal subnets for deployment. These systems primarily target cloud or edge computing. In contrast, our work addresses low-latency solutions for 3D object detection in autonomous driving tasks, ensuring latency and accuracy SLOs directly on embedded GPUs where data is generated.

7 Discussion

Generalizability of Agile3D. To achieve optimal performance, Agile3D requires offline MEF training, profiling, and CARL controller training using datasets collected from each specific dataset hardware configuration (*e.g.,* Waymo, nuScenes, and KITTI, which utilize different vehicles and LiDAR sensors). Given the available datasets, Agile3D adheres to standard machine learning practices, incurring only a one-time training cost per dataset setup. Such retraining is essential because datasets inherently vary across diverse hardware and environmental conditions (*e.g.,* vehicles, LiDAR configurations, and cities worldwide). Future work should investigate online training strategies using real-time profiling data, potentially enabling Agile3D to generalize effectively to previously unseen hardware setups and operating environments.

Evaluation under real-world scenarios. We evaluate Agile3D under synthetic contentions in a laboratory environment. Synthetic contention has been widely adopted in prior studies, including SOTA approaches such as Chanakya [15] and LiteReconfig [58], due to its effectiveness in creating reproducible training and evaluation scenarios. Future work should consider evaluating performance under realistic GPU resource-sharing conditions.

8 Conclusion

Agile3D, our adaptive 3D object detection system for embedded GPUs, excels in achieving SOTA accuracy while consistently meeting stringent runtime latency SLOs across diverse resource contention levels. By leveraging the MEF and CARL controller, Agile3D efficiently buffers all 3D models in GPU memory, enabling rapid model switching within 1 ms. The system features two complementary and innovative controllers: #1. CARL Controller: designed for high contention scenarios with significant latency ranges, combines supervised training with DPO fine-tuning. This enables it to dynamically adapt to resource and input fluctuations, ensuring optimal performance. #2. DA-LUT Controller: Optimized for contention-free scenarios, it efficiently selects execution branches with minimal overhead. Across multiple datasets and hardware platforms, Agile3D demonstrates superior adaptability and accuracy-latency trade-offs. It consistently meets latency SLOs—100-500 ms on Waymo (Orin), 100-250 ms on nuScenes (Orin), 33-75 ms on KITTI (Orin), and 50-100 ms on KITTI (Xavier)—while achieving up to +3% over adaptive controllers like Chanakya and LiteReconfig, and +7% accuracy gains over 3D detection models such as DSVT, CenterPoint, and PointPillars. With its robust performance under varying contention levels and ability to meet stringent latency constraints, Agile3D emerges as a leading solution for adaptive 3D detection on embedded GPUs.

Acknowledgments. This material is based in part upon work supported by the National Science Foundation under Grant Numbers CNS-2333491 / 2333487 (NSF Frontier) and CNS-2146449 (NSF CAREER award), and by the Army Research Lab under contract number W911NF-2020221. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the sponsors. The authors thank the reviewers and artifact evaluators for their enthusiastic comments, and the anonymous shepherd for their insightful feedback.

References

[1] AGHDAM, H. H., HERAVI, E. J., DEMILEW, S. S., AND LAGANIERE, R. Rad: Realtime and accurate 3d object detection on embedded systems. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2021), pp. 2875–2883.

[2] ARNOLD, E., AL-JARRAH, O. Y., DIANATI, M., FALLAH, S., OXTOBY, D., AND MOUZAKITIS, A. A survey on 3d object detection methods for autonomous driving applications. *IEEE Transactions on Intelligent Transportation Systems 20*, 10 (2019), 3782–3795.

[3] CAESAR, H., BANKITI, V., LANG, A. H., VORA, S., LIONG, V. E., XU, Q., KRISHNAN, A., PAN, Y., BALDAN, G., AND BEIJBOM, O. nuscenes: A multimodal dataset for autonomous driving. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (2020), pp. 11621–11631.

[4] CAI, H., GAN, C., WANG, T., ZHANG, Z., AND HAN, S. Once for all: Train one network and specialize it for efficient deployment. In *International Conference on Learning Representations* (2020).

[5] CHEN, Q., WANG, Y., YANG, T., ZHANG, X., CHENG, J., AND SUN, J. You only look one-level feature. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (2021), pp. 13039–13048.

[6] CHEN, X., MA, H., WAN, J., LI, B., AND XIA, T. Multi-view 3d object detection network for autonomous driving. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition* (2017), pp. 1907–1915.

[7] CHIN, T.-W., DING, R., AND MARCULESCU, D. Adascale: Towards real-time video object detection using adaptive scaling. *Proceedings of Machine Learning and Systems 1* (2019), 431–441.

[8] CHRISTIANO, P. F., LEIKE, J., BROWN, T., MARTIC, M., LEGG, S., AND AMODEI, D. Deep reinforcement learning from human preferences. *Advances in neural information processing systems 30* (2017).

[9] DOSOVITSKIY, A., BEYER, L., KOLESNIKOV, A., WEISSENBORN, D., ZHAI, X., UNTERTHINER, T., DEHGHANI, M., MINDERER, M., HEIGOLD, G., GELLY, S., USZKOREIT, J., AND HOULSBY, N. An image is worth 16x16 words: Transformers for image recognition at scale. *ICLR* (2021).

[10] FANG, B., ZENG, X., ZHANG, F., XU, H., AND ZHANG, M. Flexdnn: Input-adaptive on-device deep learning for efficient mobile vision. In *2020 IEEE/ACM Symposium on Edge Computing (SEC)* (2020), IEEE, pp. 84–95.

[11] FANG, B., ZENG, X., AND ZHANG, M. Nestdnn: Resource-aware multi-tenant on-device deep learning for continuous mobile vision. In *Proceedings of the 24th Annual International Conference on Mobile Computing and Networking* (2018), pp. 115–127.

[12] FENG, C., ZHONG, Y., GAO, Y., SCOTT, M. R., AND HUANG, W. Tood: Task-aligned one-stage object detection. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)* (2021), IEEE Computer Society, pp. 3490–3499.

[13] FORUM, N. D. Questions on per-process gpu utilization, 2024. Accessed: 2024-12-07.

[14] GEIGER, A., LENZ, P., AND URTASUN, R. Are we ready for autonomous driving? the kitti vision benchmark suite. In *2012 IEEE conference on computer vision and pattern recognition* (2012), IEEE, pp. 3354–3361.

[15] GHOSH, A., BALLOLI, V., NAMBI, A., SINGH, A., AND GANU, T. Chanakya: Learning runtime decisions for adaptive real-time perception. In *Advances in Neural Information Processing Systems* (2023), A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, Eds., vol. 36, Curran Associates, Inc., pp. 55668–55680.

[16] GU, A., AND DAO, T. Mamba: Linear-time sequence modeling with selective state spaces, 2024.

[17] GUJARATI, A., KARIMI, R., ALZAYAT, S., HAO, W., KAUFMANN, A., VIGFUSSON, Y., AND MACE, J. Serving {DNNs} like clockwork: Performance predictability from the bottom up. In *14th USENIX Symposium on Operating Systems Design and Implementation (OSDI 20)* (2020), pp. 443–462.

[18] HAN, K., WANG, Y., TIAN, Q., GUO, J., XU, C., AND XU, C. Ghostnet: More features from cheap operations. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (2020), pp. 1580–1589.

[19] HOWARD, A., SANDLER, M., CHU, G., CHEN, L.-C., CHEN, B., TAN, M., WANG, W., ZHU, Y., PANG, R., VASUDEVAN, V., ET AL. Searching for mobilenetv3. In *Proceedings of the IEEE/CVF international conference on computer vision* (2019), pp. 1314–1324.

[20] JIANG, A. H., WONG, D. L.-K., CANEL, C., TANG, L., MISRA, I., KAMINSKY, M., KOZUCH, M. A., PILLAI, P., ANDERSEN, D. G., AND GANGER, G. R. Mainstream: Dynamic {Stem-Sharing} for {Multi-Tenant} video processing. In *2018 USENIX Annual Technical Conference (USENIX ATC 18)* (2018), pp. 29–42.

[21] JIANG, J., ANANTHANARAYANAN, G., BODIK, P., SEN, S., AND STOICA, I. Chameleon: scalable adaptation of video analytics. In *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication* (2018), pp. 253–266.

[22] KARUMBUNATHAN, L. S. Nvidia jetson agx orin series, 2022.

[23] LANG, A. H., VORA, S., CAESAR, H., ZHOU, L., YANG, J., AND BEIJBOM, O. Pointpillars: Fast encoders for object detection from point clouds. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (2019), pp. 12697–12705.

[24] LI, Y., MA, L., ZHONG, Z., LIU, F., CHAPMAN, M. A., CAO, D., AND LI, J. Deep learning for lidar point clouds in autonomous driving: A review. *IEEE Transactions on Neural Networks and Learning Systems 32*, 8 (2020), 3412–3432.

[25] LIN, J., CHEN, W.-M., LIN, Y., GAN, C., HAN, S., ET AL. Mcunet: Tiny deep learning on iot devices. *Advances in Neural Information Processing Systems 33* (2020), 11711–11722.

[26] LIU, W., ANGUELOV, D., ERHAN, D., SZEGEDY, C., REED, S., FU, C.-Y., AND BERG, A. C. Ssd: Single shot multibox detector. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I 14* (2016), Springer, pp. 21–37.

[27] MACGLASHAN, J., HO, M. K., LOFTIN, R., PENG, B., WANG, G., ROBERTS, D. L., TAYLOR, M. E., AND LITTMAN, M. L. Interactive learning from policy-dependent human feedback. In *International conference on machine learning* (2017), PMLR, pp. 2285–2294.

[28] MAO, J., SHI, S., WANG, X., AND LI, H. 3d object detection for autonomous driving: A comprehensive survey. *International Journal of Computer Vision 131*, 8 (2023), 1909–1963.

[29] MENG, Y., LIN, C.-C., PANDA, R., SATTIGERI, P., KARLINSKY, L., OLIVA, A., SAENKO, K., AND FERIS, R. Ar-net: Adaptive frame resolution for efficient action recognition. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part VII 16* (2020), Springer, pp. 86–104.

[30] NIGADE, V., BAUSZAT, P., BAL, H., AND WANG, L. Jellyfish: Timely inference serving for dynamic edge networks. In *2022 IEEE Real-Time Systems Symposium (RTSS)* (2022), IEEE, pp. 277–290.

[31] OUYANG, L., WU, J., JIANG, X., ALMEIDA, D., WAINWRIGHT, C., MISHKIN, P., ZHANG, C., AGARWAL, S., SLAMA, K., RAY, A., ET AL. Training language models to follow instructions with human feedback. *Advances in neural information processing systems 35* (2022), 27730–27744.

[32] QI, C. R., SU, H., MO, K., AND GUIBAS, L. J. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2017), pp. 652–660.

[33] QI, C. R., YI, L., SU, H., AND GUIBAS, L. J. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in neural information processing systems* (2017), p. 5105–5114.

[34] QIN, D., LEICHNER, C., DELAKIS, M., FORNONI, M., LUO, S., YANG, F., WANG, W., BANBURY, C., YE, C., AKIN, B., ET AL. Mobilenetv4: Universal models for the mobile ecosystem. In *European Conference on Computer Vision* (2025), Springer, pp. 78–96.

[35] RAFAILOV, R., SHARMA, A., MITCHELL, E., MANNING, C. D., ERMON, S., AND FINN, C. Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems 36* (2024).

[36] RAN, X., CHEN, H., ZHU, X., LIU, Z., AND CHEN, J. Deepdecision: A mobile deep learning framework for edge video analytics. In *IEEE INFOCOM 2018-IEEE Conference on Computer Communications* (2018), IEEE, pp. 1421–1429.

[37] RAO, Y., ZHAO, W., LIU, B., LU, J., ZHOU, J., AND HSIEH, C.-J. Dynamicvit: Efficient vision transformers with dynamic token sparsification. *Advances in neural information processing systems 34* (2021), 13937–13949.

[38] REN, S., HE, K., GIRSHICK, R., AND SUN, J. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems 28* (2015).

[39] RIQUELME, C., PUIGCERVER, J., MUSTAFA, B., NEUMANN, M., JENATTON, R., SUSANO PINTO, A., KEYSERS, D., AND HOULSBY, N. Scaling vision with sparse mixture of experts. *Advances in Neural Information Processing Systems 34* (2021), 8583–8595.

[40] ROMERO, F., LI, Q., YADWADKAR, N. J., AND KOZYRAKIS, C. {INFaaS}: Automated model-less inference serving. In *2021 USENIX Annual Technical Conference (USENIX ATC 21)* (2021), pp. 397–411.

[41] SCHULMAN, J., WOLSKI, F., DHARIWAL, P., RADFORD, A., AND KLIMOV, O. Proximal policy optimization algorithms, 2017.

[42] SHI, S., GUO, C., JIANG, L., WANG, Z., SHI, J., WANG, X., AND LI, H. Pv-rcnn: Point-voxel feature set abstraction for 3d object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2020), pp. 10529–10538.

[43] SHI, S., WANG, Z., SHI, J., WANG, X., AND LI, H. From points to parts: 3d object detection from point cloud with part-aware and part-aggregation network. *IEEE transactions on pattern analysis and machine intelligence 43*, 8 (2020), 2647–2664.

[44] SUALEH, M., AND KIM, G.-W. Visual-lidar based 3d object detection and tracking for embedded systems. *IEEE Access 8* (2020), 156285–156298.

[45] SUN, P., KRETZSCHMAR, H., DOTIWALLA, X., CHOUARD, A., PATNAIK, V., TSUI, P., GUO, J., ZHOU, Y., CHAI, Y., CAINE, B., ET AL. Scalability in perception for autonomous driving: Waymo open dataset. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (2020), pp. 2446–2454.

[46] SUN, P., ZHANG, R., JIANG, Y., KONG, T., XU, C., ZHAN, W., TOMIZUKA, M., LI, L., YUAN, Z., WANG, C., ET AL. Sparse r-cnn: End-to-end object detection with learnable proposals. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (2021), pp. 14454–14463.

[47] TAN, M., CHEN, B., PANG, R., VASUDEVAN, V., SANDLER, M., HOWARD, A., AND LE, Q. V. Mnasnet: Platform-aware neural architecture search for mobile. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2019), pp. 2820–2828.

[48] TAN, M., AND LE, Q. Efficientnet: Rethinking model scaling for convolutional

neural networks. In *International conference on machine learning* (2019), PMLR, pp. 6105–6114.

[49] Tang, C., Zhang, L. L., Jiang, H., Xu, J., Cao, T., Zhang, Q., Yang, Y., Wang, Z., and Yang, M. Elasticvit: Conflict-aware supernet training for deploying fast vision transformer on diverse mobile devices. In *Proceedings of the IEEE/CVF International Conference on Computer Vision* (2023), pp. 5829–5840.

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

[51] Teerapittayanon, S., McDanel, B., and Kung, H.-T. Branchynet: Fast inference via early exiting from deep neural networks. In *2016 23rd International Conference on Pattern Recognition (ICPR)* (2016), IEEE, pp. 2464–2469.

[52] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. *Advances in neural information processing systems 30* (2017).

[53] Wang, H., Shi, C., Shi, S., Lei, M., Wang, S., He, D., Schiele, B., and Wang, L. Dsvt: Dynamic sparse voxel transformer with rotated sets. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2023), pp. 13520–13529.

[54] Wang, H., Wu, Z., Liu, Z., Cai, H., Zhu, L., Gan, C., and Han, S. Hat: Hardware-aware transformers for efficient natural language processing. In *Annual Conference of the Association for Computational Linguistics* (2020).

[55] Wu, B., Dai, X., Zhang, P., Wang, Y., Sun, F., Wu, Y., Tian, Y., Vajda, P., Jia, Y., and Keutzer, K. Fbnet: Hardware-aware efficient convnet design via differentiable neural architecture search. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2019), pp. 10734–10742.

[56] Wu, Y., Wang, Y., Zhang, S., and Ogai, H. Deep 3d object detection networks using lidar data: A review. *IEEE Sensors Journal 21*, 2 (2020), 1152–1171.

[57] Xu, R., Kumar, R., Wang, P., Bai, P., Meghanath, G., Chaterji, S., Mitra, S., and Bagchi, S. Approxnet: Content and contention-aware video object classification system for embedded clients. *ACM Transactions on Sensor Networks (TOSN) 18*, 1 (2021), 1–27.

[58] Xu, R., Lee, J., Wang, P., Bagchi, S., Li, Y., and Chaterji, S. Litereconfig: cost and content aware reconfiguration of video object detection systems for mobile gpus. In *Proceedings of the Seventeenth European Conference on Computer Systems* (2022), pp. 334–351.

[59] Xu, R., Zhang, C.-l., Wang, P., Lee, J., Mitra, S., Chaterji, S., Li, Y., and Bagchi, S. Approxdet: content and contention-aware approximate object detection for mobiles. In *Proceedings of the 18th Conference on Embedded Networked Sensor Systems* (2020), pp. 449–462.

[60] Yan, Y., Mao, Y., and Li, B. Second: Sparsely embedded convolutional detection. *Sensors 18*, 10 (2018), 3337.

[61] Yang, B., Luo, W., and Urtasun, R. Pixor: Real-time 3d object detection from point clouds. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition* (2018), pp. 7652–7660.

[62] Yang, H., He, T., Liu, J., Chen, H., Wu, B., Lin, B., He, X., and Ouyang, W. Gd-mae: generative decoder for mae pre-training on lidar point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2023), pp. 9403–9414.

[63] Yang, J., Shi, S., Ding, R., Wang, Z., and Qi, X. Towards efficient 3d object detection with knowledge distillation. *Advances in Neural Information Processing Systems 35* (2022), 21300–21313.

[64] Yin, T., Zhou, X., and Krahenbuhl, P. Center-based 3d object detection and tracking. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (2021), pp. 11784–11793.

[65] Zhang, H., Ananthanarayanan, G., Bodik, P., Philipose, M., Bahl, P., and Freedman, M. J. Live video analytics at scale with approximation and {Delay-Tolerance}. In *14th USENIX Symposium on Networked Systems Design and Implementation (NSDI 17)* (2017), pp. 377–392.

[66] Zhang, H., Chang, H., Ma, B., Wang, N., and Chen, X. Dynamic r-cnn: Towards high quality object detection via dynamic training. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XV 16* (2020), Springer, pp. 260–275.

[67] Zhang, X., Zhou, X., Lin, M., and Sun, J. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2018), pp. 6848–6856.

[68] Zhou, Y., Sun, P., Zhang, Y., Anguelov, D., Gao, J., Ouyang, T., Guo, J., Ngiam, J., and Vasudevan, V. End-to-end multi-view fusion for 3d object detection in lidar point clouds. In *Conference on Robot Learning* (2020), PMLR, pp. 923–932.

[69] Zhou, Y., and Tuzel, O. Voxelnet: End-to-end learning for point cloud based 3d object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (2018), pp. 4490–4499.

[70] Zhu, X., Ma, Y., Wang, T., Xu, Y., Shi, J., and Lin, D. Ssn: Shape signature networks for multi-class object detection from point clouds. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings* (2020), Springer, pp. 581–597.