Rightsizing ML: From Sparse 3D Autonomy to Semi-Supervised Insights

Prof. Somali Chaterji
Associate Professor
Agricultural and Biological Engineering;
Electrical & Computer Engineering
Purdue University

Outline

- Broad Themes in ICAN
- 3D Object Detection on Mobile GPU [Mobisys-25, ACM-TODAES-23, EuroSys-22, CVPR-22, SenSys-20]
- Semi-Supervised Segmentation [CVPR-25]
- Takeaways

The Two Thrusts of ICAN

Thrust 1: Intelligence with Internet-of-(Small)-Things (IoST)

- **Context- and Resource-Aware Computing:** We trim neural networks at the sensor level, optimizing accuracy while adapting to resource constraints.
- **Middleware and Edge Computing:** We design middleware to enable seamless computation across sensors and edge devices, for low-latency, real-time performance.
- **Cloud NoSQL Database:** Through KeyByte, we tune databases automatically and in the background for performance while staying within cost constraints.

Impact

1st instantiation of streaming video object detection and LiDAR object detection on NVIDIA Mobile GPUs
1st edge device-cloud decomposition of video human activity recognition for low-bandwidth satellite networks, demo in NSF CPS 2025 PI meeting
1st solution for dynamically tuning NoSQL DBs when application characteristics change

The Two Thrusts of ICAN

Thrust 2: Computational Genomics and One Health

- **Efficient Algorithm Evolution:** We develop reusable software modules to accelerate genomic analysis and scale with growing datasets.
- **Interpretable Clustering:** We uncover new cell groups and gene signatures to enhance understanding of disease and cell function.
- **MicroRNA-based Therapies:** We design precision therapeutics targeting disease-specific pathways using regulatory RNA insights.

Impact

ACM BCB best paper award 2015 (MicroRNA therapeutics)
*ACM Sigmetrics **best paper** 2022 (serverless cloud computing databases - Orion and WiseFuse), patented with Microsoft Research*
NIH R01: my first grant as faculty (databases for metagenomics) - KeyByte, 2 awarded patents (Sophia, OptimusCloud)
1st DSL for genomics (SARVAVID: A DSL for Developing Scalable Computational Genomics Applications)

Common Thread across My Work

- **Applying machine learning to make sense of noisy, high-dimensional data for the greater good.**
- **Through Physical AI:** improving real-world systems like autonomous vehicles
- **Or through One Health:** advancing precision therapeutics by making sense of genomics data

Agile3D: Adaptive Contention- and Content-Aware 3D Object Detection for Embedded GPUs

MobiSys 2025, code available (ACM badges), patent pending

https://schaterji.io/publications/2025/agile3d/

Outline - PART 1 - Background

- Point Cloud Data Representations and Benchmark Datasets
- RLHF/PPO and DPO
- Example SOTA Algorithms for 3D (different data encoders)
    - VoxelNet: End-to-end learning for point cloud-based 3D object detection [CVPR'18]
    - PointPillars: Fast encoders for object detection from point clouds [CVPR'19]
    - CenterPoint: Center-based 3D object detection and tracking [CVPR'21]
- Resource-Constrained Hardware - Mobile GPUs

Point Cloud Data

What is Point Cloud Data?

Captured by LiDAR sensors (typically operating at 10–20 Hz)

Each frame contains a set of 3D points (x, y, z, intensity) representing the surface of objects in the environment

Key Characteristics:

Naturally sparse and non-uniform in density (needs data encoders)

Efficient and lightweight for 3D perception tasks (Lacks color, texture, and shading—unlike images)

Why Use Point Clouds?

Enables accurate 3D mapping, object detection, and localization

Applications of Point Cloud Data

Point clouds are crucial in a range of real-world applications:

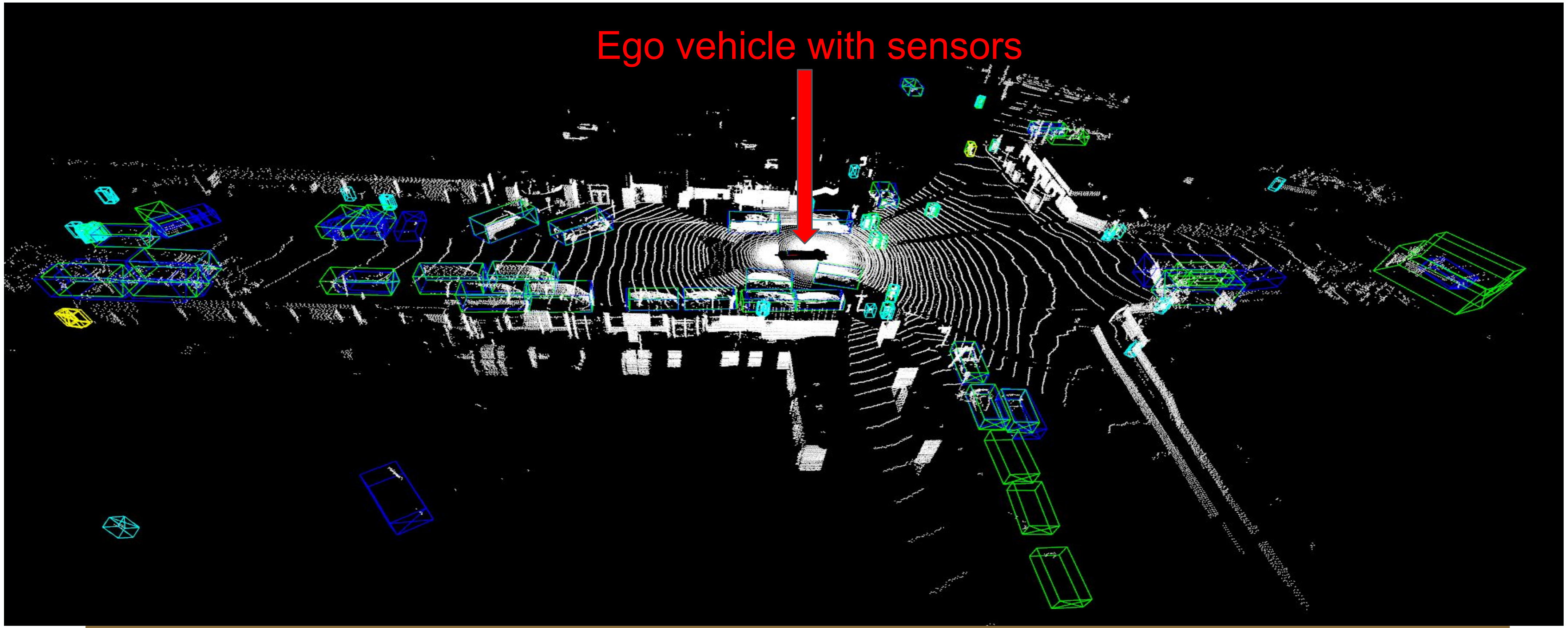
🚗 *Autonomous Vehicles* – for environment perception and obstacle avoidance

🚁 *Drones* – for terrain mapping and navigation

🤖 *Robotics* – for object manipulation and Simultaneous Localization and Mapping (SLAM)

🕶️ *AR/VR* – for real-time 3D scene understanding and interaction

Point Clouds in the Waymo Dataset

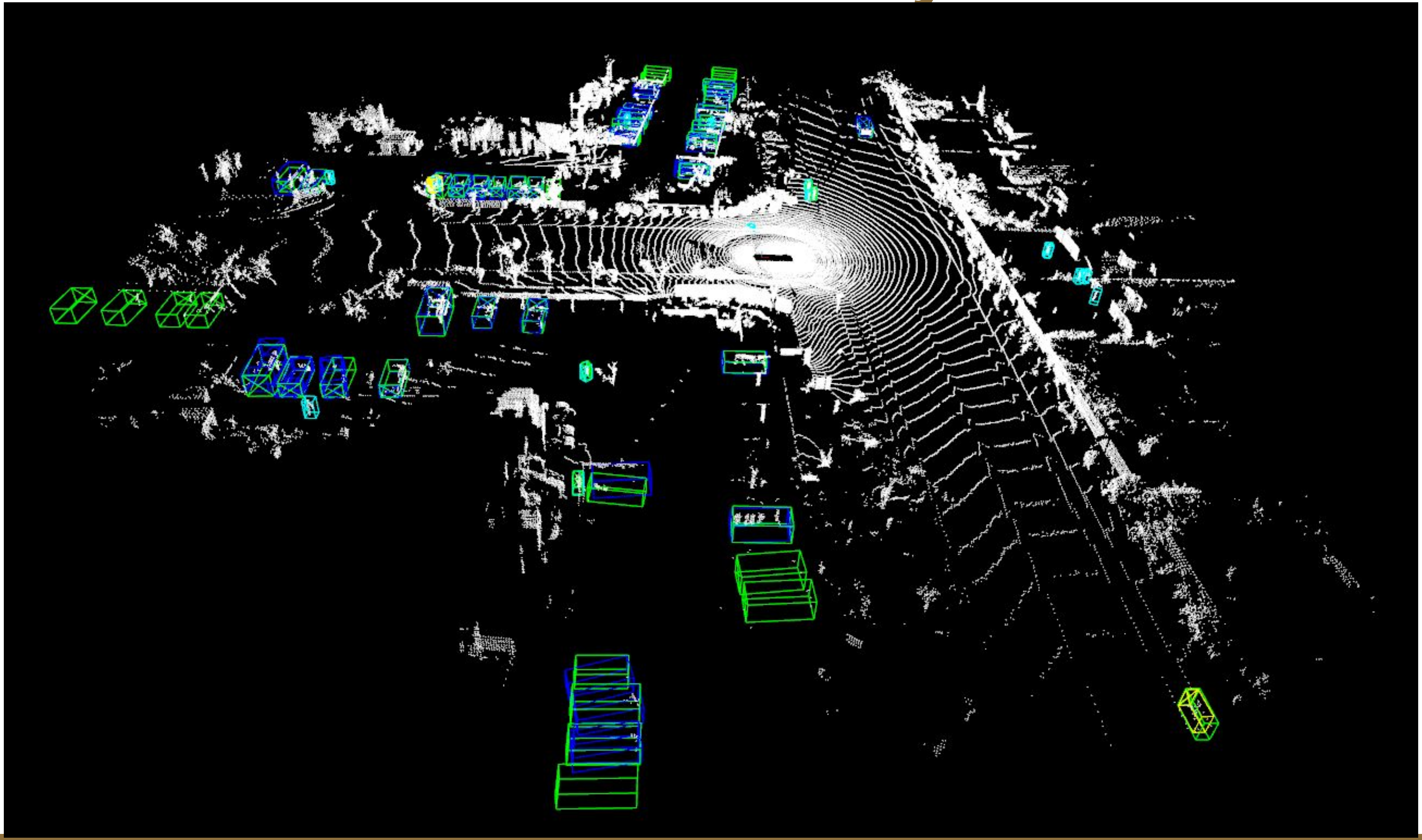


Ground Truth: **green**, vehicle: **blue**, pedestrian: **cyan**, cyclist: **yellow**

The ego vehicle, located at the center, scans the environment at 10–20 Hz in all directions, generating precise depth information.

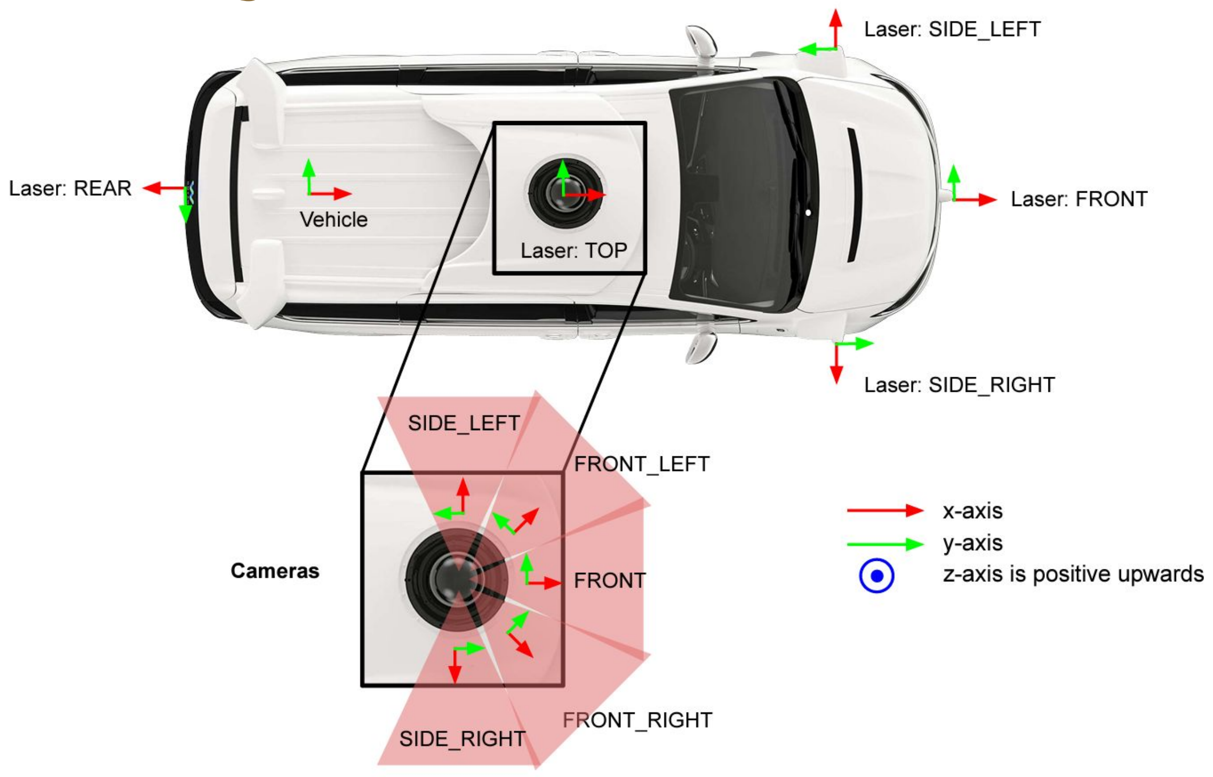
Sun, Pei, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo et al. "Scalability in perception for autonomous driving: Waymo open dataset." In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp. 2446-2454. 2020.

Point Clouds in the Waymo Dataset



Ground Truth: **green**, vehicle: **blue**, pedestrian: **cyan**, cyclist: **yellow**

Waymo Open Datasets



AVs are equipped with an increasing number of sensors, each generating high-density, multimodal data. Applications:

- 2D object detection (from cameras),
- 3D object detection (from LiDAR point clouds),
- Voice recognition (from microphones),

These sensors are often co-located on the same hardware.

As the volume and complexity of sensor data grow, and multiple AI workloads compete for limited compute resources, the need for intelligent contention-aware algorithms becomes critical.

These complex and evolving runtime environments demand more efficient, adaptive scheduling algorithms to ensure real-time performance and system robustness.

Environment Perception Sensors:

- 5x RGB Cameras
- 5x LiDAR (Light Detection and Ranging)

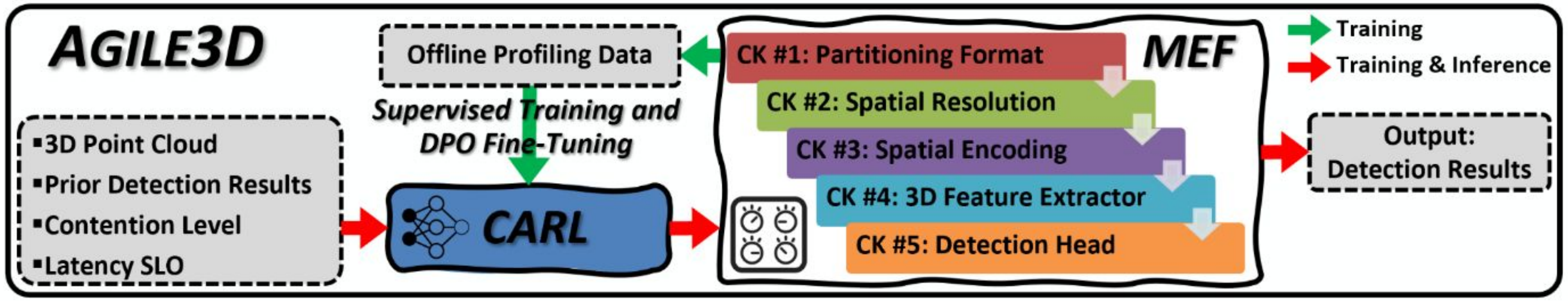
Dataset size: 1.8TB, 1150 scenes

The number of sensors and the information density of data collected from each sensor are constantly increasing, demanding more efficient algorithms.

Sun, Pei, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo et al. "Scalability in perception for autonomous driving: Waymo open dataset." In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp. 2446-2454. 2020.

Agile3D's Pipeline = MEF (dynamism) +CARL

```
Input Point Cloud                              |
(Raw LiDAR data: x, y, z coordinates)     |
                    ↓
Preprocessing & Feature Extraction     |
1. Voxelization: Divide into 3D cubes (voxels) |
2. Pillarization: Divide into vertical columns  |
                    ↓
Multi-Branch Execution Framework  (MBEF)
 Branch1|    | Branch2|    | Branch3|    | BranchN|
 (Voxel)|    | (Pillar|     | (Hybrid|    | (...)   |   |
 High    |    | Low      |   | Medium)|   |          |   |
 Acc.    |    | Latency|    |            |   |          |   |
     ↓        ↓         ↓         ↓
Dynamic Branch Selection (CARL)       |
1. Contention Monitor: Checks GPU load.        |
2. Content Analyzer: Assesses input density.   |
3. DPO Optimizer: Picks branch balancing:       |
   - Accuracy (e.g., voxel branch for dense scenes)|
   - Latency (e.g., pillar branch under contention)|
                    ↓
Output: 3D Detection Results              |
(Bounding boxes for cars, pedestrians, cyclists)  |
```



Configurations: The 5 Control Knobs (CKs)

Agile3D's flexibility comes from **five tunable parameters** that define each branch:

Control Knob	Function	Example Options
CK #1	Partitioning Format	Voxel, Pillar, Hybrid.
CK #2	Spatial Resolution	Voxel size (0.1m, 0.3m), pillar grid.
CK #3	Spatial Encoding	Sparse convolutions, dense convolutions.
CK #4	3D Feature Extractor	DSVT (Transformers), PointPillars (2D CNNs).
CK #5	Detection Head	CenterPoint, Part-A2.

Example Configuration:

- **Branch A**: Voxel (0.1m) + DSVT + CenterPoint → High accuracy, slow.
- **Branch B**: Pillar (0.2m) + PointPillars → Low latency, less detailed.

Configurations: Five Adjustable Parameters

Why do we Need to Approximate 3D Analytics

DSVT (Dynamic Sparse Voxel Transformer): Requires 13 TFLOPs to run optimally.

Embedded GPU Limitations:

NVIDIA Orin: 5.3 TFLOPs (≈41% of DSVT's requirement).

NVIDIA Xavier: 1.4 TFLOPs (≈11% of DSVT's requirement).

Consequences Without Optimization:

Latency Explosion: DSVT would take 2–10× longer to process data on these GPUs, violating real-time SLOs (*e.g.*, >500 ms).

Energy Waste: More computations = higher power draw, draining batteries in AVs.

Hardware Overload: Pushing GPUs beyond their TFLOPs capacity risks thermal throttling or crashes.

Wang, Haiyang, Chen Shi, Shaoshuai Shi, Meng Lei, Sen Wang, Di He, Bernt Schiele, and Liwei Wang. "DSVT: Dynamic sparse voxel transformer with rotated sets." In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 13520-13529. 2023.

How does Agile3D Approximate?

Agile3D uses adaptive strategies to reduce computational demands while preserving accuracy

Dynamic Branch Selection:

CARL Controller: Chooses between high-accuracy branches (*e.g.*, DSVT) and low-latency branches (e.g., CP) based on real-time contention.

Example: In high-contention scenarios, prioritize CP branches (low TFLOPs) to stay within GPU limits.

Voxel/Pillar Optimization:

Adjusts voxel sizes to balance resolution and computational cost (e.g., larger voxels = fewer computations).

Hybrid Training (SL+DPO):

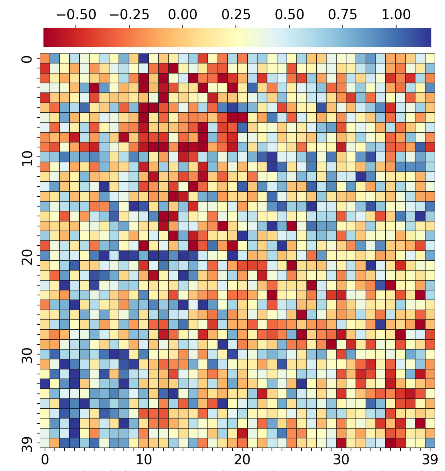
Trains models to maximize accuracy within hardware constraints, avoiding brute-force computation.

Example Workflow:

Dense urban area with pedestrians → Use a DSVT model trained on small voxels (0.1m).

Highway with distant cars → Switch to a PointPillars model trained on large voxels (0.3m).

To avoid retraining on the fly, *Agile3D* pre-trains multiple models, each optimized for a specific voxel size or backbone network. At runtime, it switches between these models based on the scene's needs.



Switching Overhead on Orin (ms)

Switching overhead: Agile3D buffers all MEF branches in memory, using <8GB of RAM, well below the memory capacity of modern GPUs. This design will introduce a minor branch-switching overhead. Pre-buffering limits overhead to under 1 ms, as transitions only require memory-to-GPU operations. In contrast, loading models from disk causes latency spikes exceeding 200 ms. Disk-to-GPU 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). Y-axis: source and X-axis: destination branches.

Agile3D’s Multi-Model Approach: Adapting to the Scene (Content) and Resource Contention

Why Multiple Models are Necessary? *Single- versus cross-model branching.*

Varied Voxel/Pillar Resolutions: Different scenes require different levels of detail.

For example, nearby pedestrian detection benefits from smaller voxels and finer resolution, while car detection across a larger area works better with larger voxels and broader coverage.

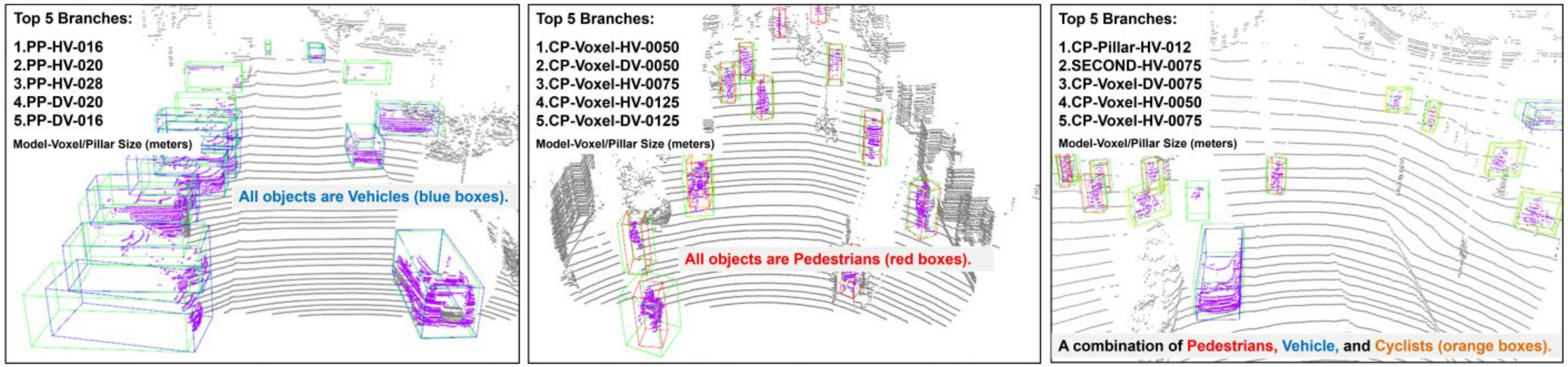
Diverse Backbone Networks: Different NN architectures have varying strengths.

Sparse 3D CNNs (like SECOND), pillar-based 2D CNNs (like PointPillars), and Transformers (like DSVT) each perform best in specific scenarios, such as densely populated areas versus simpler highway environments.

Retraining Requirements for Parameter Adjustments: In 2D image processing, image size can be adjusted on the fly.

In 3D, changing voxel size or detection range usually disrupts the model’s architecture and necessitates retraining. *Agile3D* addresses this by pre-training multiple configurations, enabling runtime switching without the need for retraining.

Motivation: Need for a Content-Aware Design



Context-Aware Adaptation in 3D Object Detection via Model Selection: Empirical results demonstrate the necessity for content-adaptive model selection based on distinct point cloud characteristics: [L] vehicle-only, [M] pedestrian-only, and [R] mixed pedestrian, cyclist, and vehicle scenes, revealing no single model is universally optimal.

The variability of top branches across distinct point clouds requires Content-Aware Design:

- In vehicle-only scenarios ([L]), Pillar-based models excel, showcasing their suitability for less complex contexts.
- For pedestrian-only scenes ([M]), CP-SEC models are superior, indicating their efficiency in detecting smaller objects with complex orientions.
- The mixed context ([R]) presents a mix of CP-PP, SECOND, and CP-SEC models, indicating the complexity of model selection in diverse environments.

Bridging 2D–3D Differences to an Agile Multi-Model Approach

Recap of 2D vs. 3D Key Differences

- 3D LiDAR data is sparse and irregular, demanding an extra data encoder (voxel/pillar).
- Memory usage is lower for 3D models (sparse ops) but computationally more complex at runtime.
- Occlusions are more severe in 3D (truly missing points) compared to 2D video occlusions.

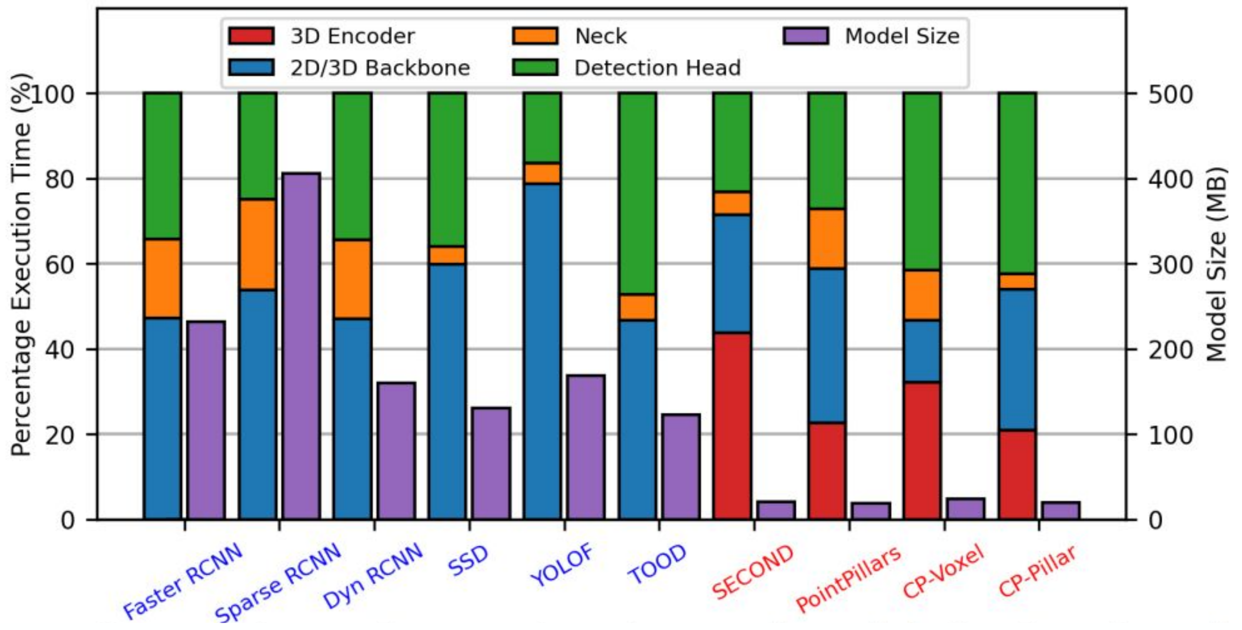
Why Simple Adaptation from 2D Falls Short

- Directly tuning image size (2D) ≠ adjusting voxel size (3D), which usually breaks the model and requires retraining.
- Large, uneven detection ranges in 3D must be carefully managed to handle near/far objects with different configurations.

Motivation for Multiple Pre-Trained 3D Models

- Each model uses distinct voxel/pillar settings or backbones (SECOND, PointPillars, etc.).
- We can store multiple 3D models in memory, thanks to their smaller parameter footprints.
- This lets our runtime system (*Agile3D*) instantly switch to the best-suited model for the current scene and latency SLO.

Motivation: 2D vs 3D Object Detection



Comparison of execution time distribution and model size between 2D and 3D object detection models. While 3D models demand higher computational resources for processing point clouds, they exhibit significantly better memory efficiency, with an average model size of 20.53 MB compared to 203.32 MB for 2D models.

Techniques from adaptive 2D object detection are insufficient for 3D:

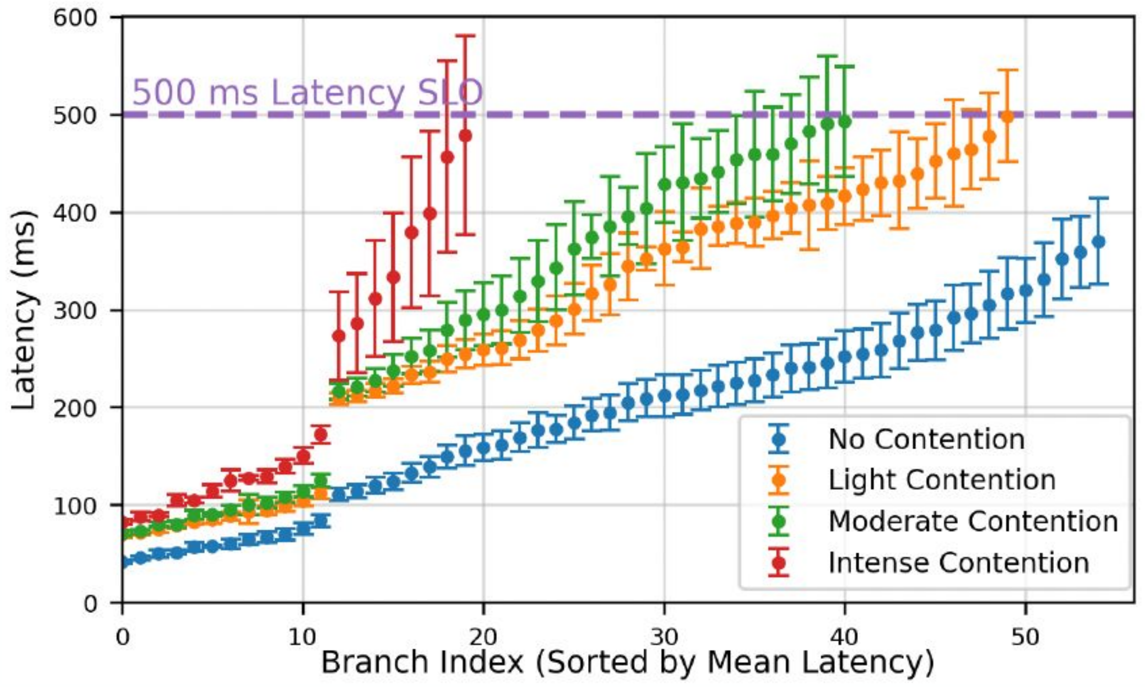
- Critical Role of 3D Encoder
- Difference in Memory Consumption
  - 3D data is more efficient representation
  - 2D data has a lot of noise and background
- Interdependencies in System Design

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 2D techniques when applied to 3D systems, as 3D models require specialized encoders to process point clouds into structured spatial features.

Mobile GPUs: Limited Compute and Memory

Feature	**NVIDIA Jetson AGX Xavier**	**NVIDIA Jetson AGX Orin**
Price	$699	$1,999
CPU	8-core NVIDIA Carmel Armv8.2 64-bit	12-core Arm Cortex-A78AE v8.2 64-bit
	8MB L2 + 4MB L3 Cache	3MB L2 + 6MB L3 Cache
	Max Frequency: 2265MHz	Max Frequency: 2200MHz
GPU	512-core NVIDIA Volta with 64 Tensor Cores	2048-core NVIDIA Ampere with 64 Tensor Cores
	Max Frequency: 1377MHz	Max Frequency: 1300MHz
Memory	32GB 256-bit LPDDR4x	64GB 256-bit LPDDR5
	Bandwidth: 136.5GB/s	Bandwidth: 204.8GB/s
	Max Frequency: 2133MHz	Max Frequency: 3200MHz
Storage	32GB eMMC 5.1 + 1TB SSD	64GB eMMC 5.1 + 2TB SSD
Power	8 Modes: 10W/15W/30W	5 Modes: 15W-40W
AI Performance	32 TOPS	275 TOPS

Motivation: High Latency Variance of 3D Models under Contention



Two distinct latency variances are observed:

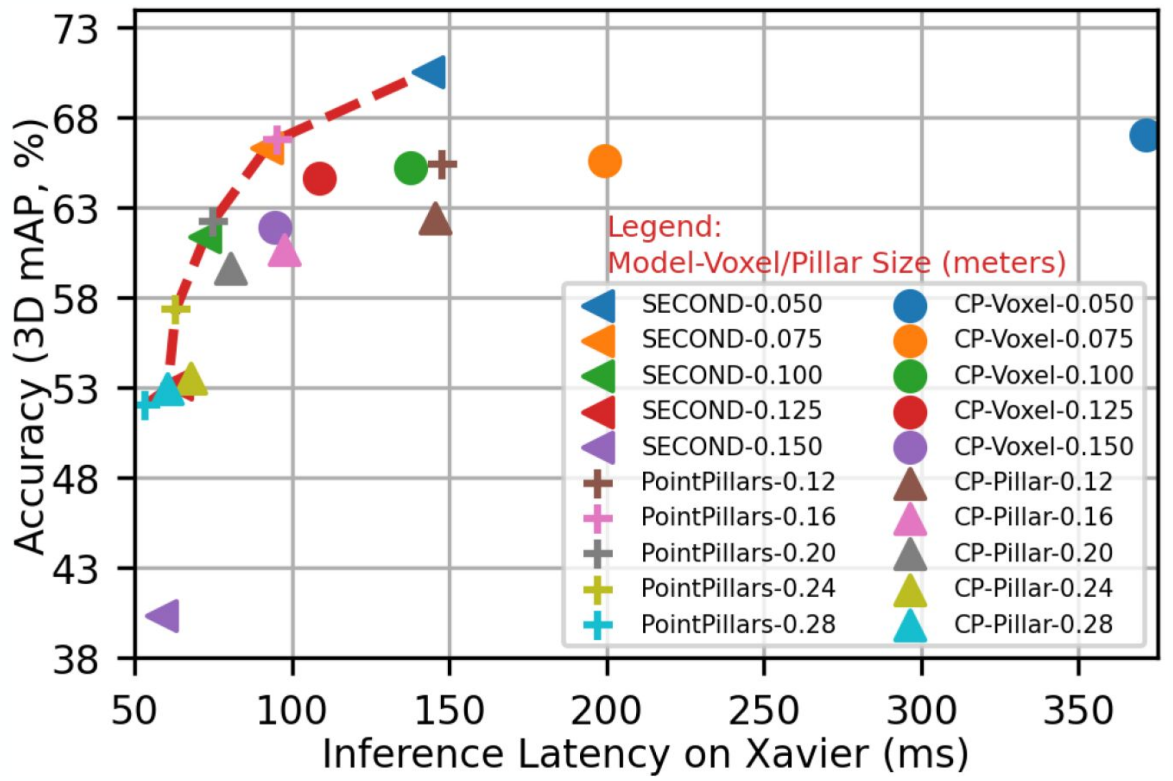
Within-Model Variance:
Due to input density differences—dense point clouds require higher computational effort, increasing latency, especially under contention. A branch processing a dense urban scene might take 500 ms under contention vs. 300 ms for a sparse scene.

Between-Branch Variance: From architectural differences across branches (model configurations)
Operations like voxelization, grouping, sampling, and sparse/dense convolutions introduce variable computational demands, further amplifying latency variability under resource contention. A transformer-based branch might take 600 ms under contention, while a pillar-based branch takes 350 ms.

Mean Latency with Standard Deviation (SD) across Branches.
Higher resource contention increases latency variability, limiting the number of feasible branches within the critical 500 ms SLO. This emphasizes the importance of designing contention- and content-aware controls for reliable real-time 3D inference.

Model configurations
- Voxelization Methods: Sparse *vs*. dense convolutions.
- Feature Extractors: Transformers (DSVT) vs. 2D CNNs (PointPillars).
- Detection Heads: CenterPoint vs. Part-A2.

Motivation: Need for a Multi-Model Design



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.

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.

We evaluated four popular 3D models each with five distinct voxel/pillar sizes.

No single model consistently occupies the Pareto frontier under all conditions.

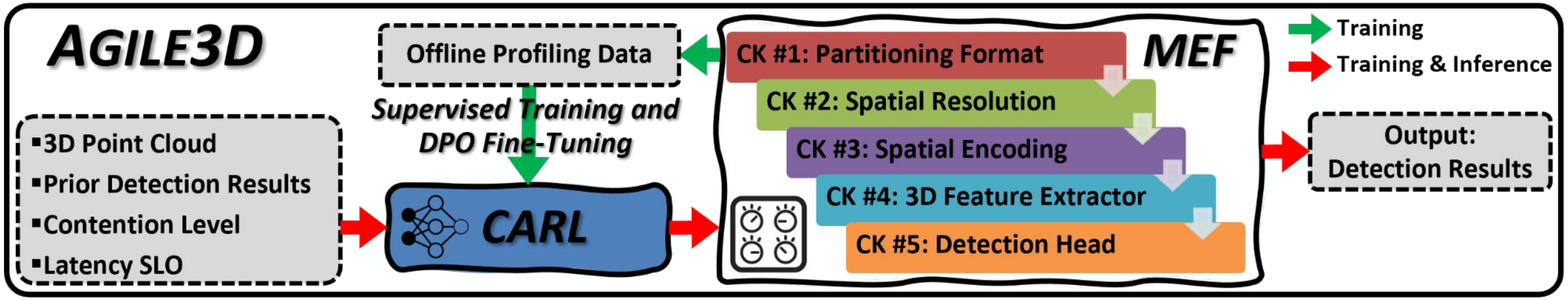
The Gap

No 3D object detection method on mobile GPUs to adapt at runtime to different input complexities and hardware resource contentions

Four Key Challenges:

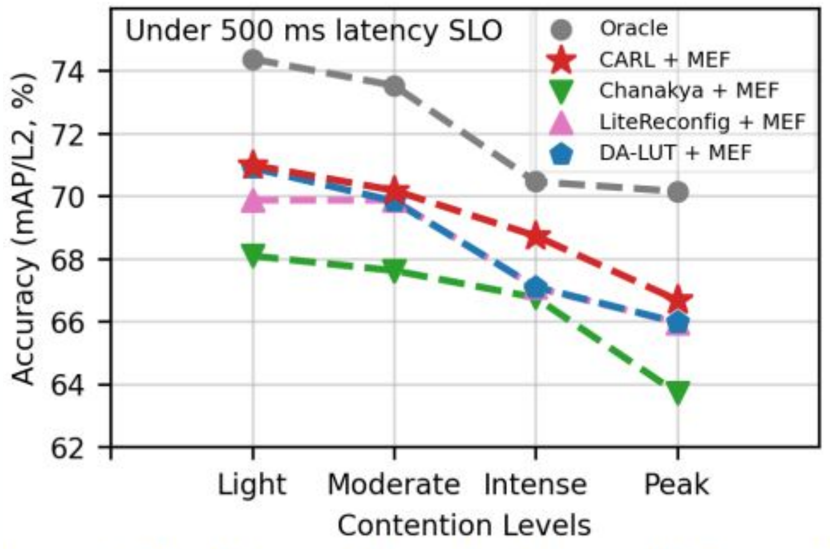
1. **Techniques from adaptive 2D systems are inadequate for 3D**
2. **Inflexibility of the 3D Models**
3. **Interdependencies in System Design**
4. **Necessity for Contention- and Content-Aware Design**

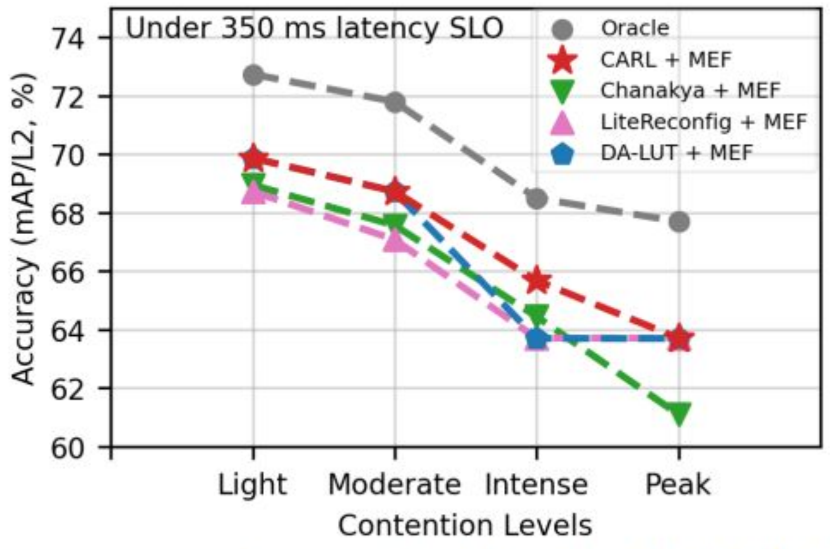
Overview of Agile3D

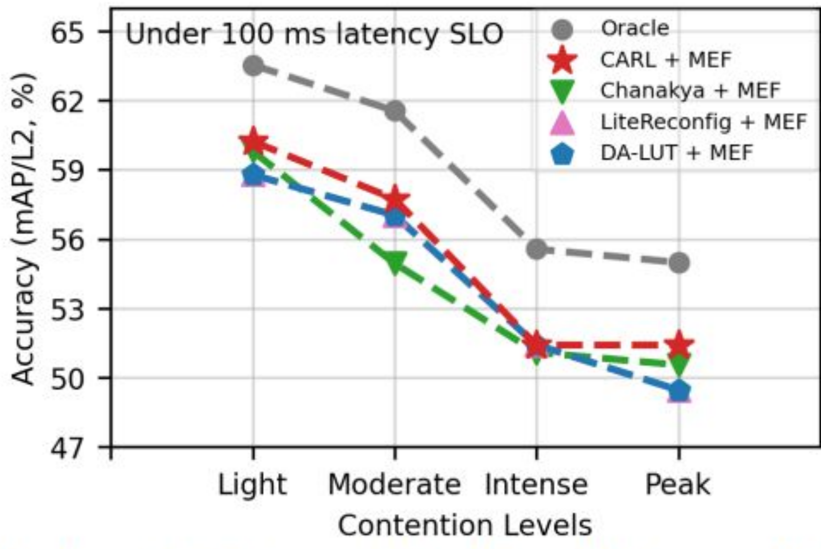


- Context-Aware Reinforcement Learning (CARL) Scheduler achieved by three steps
    - Offline Profiling to collect the latency, accuracy, and detection results
    - Initial training with supervised learning using the offline profiling data
    - Fine Tuning with Direct Preference Optimization (DPO)
        - DPO is a method to fine-tune Large language models (LLMs) to align with human preferences.
        - Traditionally, DPO requires humans to label 'good answers' versus 'bad answers'
        - However, we employ an Oracle Scheduler based on beam search to label the optimal branch.
- Multi-Branch Execution Framework (MEF)
    - Five control knobs lead to 50+ branches that can cover a wide range of latency and accuracy
    - Four out of the Five control knobs are specifically designed for 3D tasks

Evaluation: Agile3D under Varying Contention Levels and Different Latency SLOs

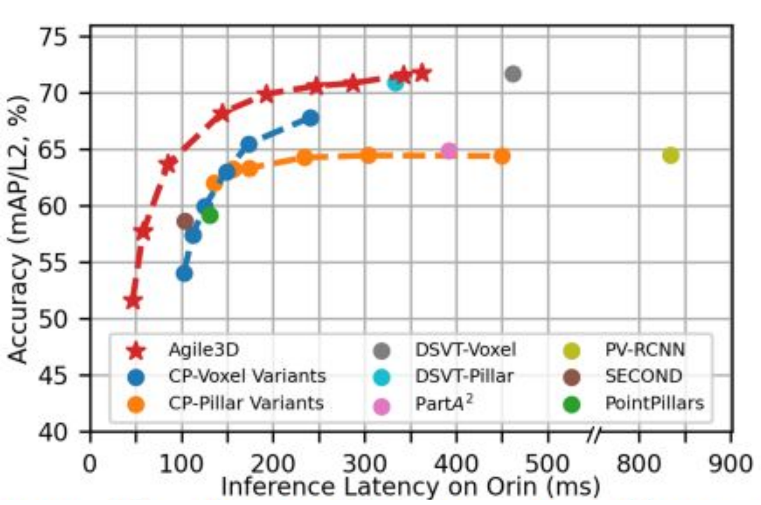


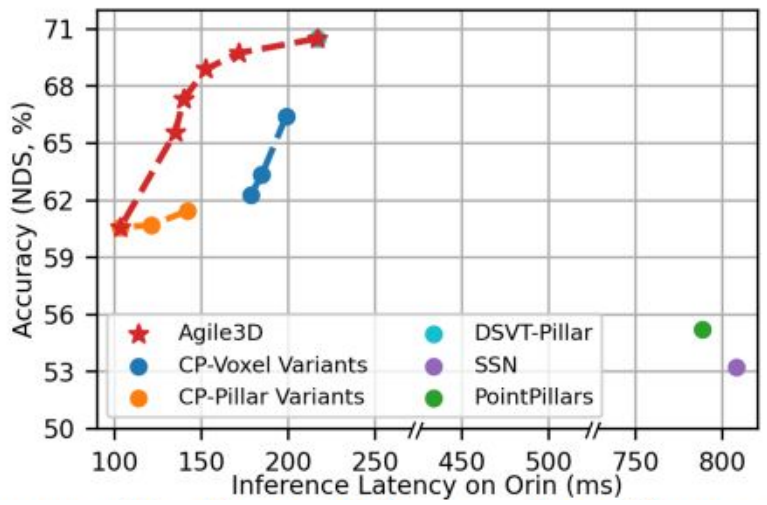


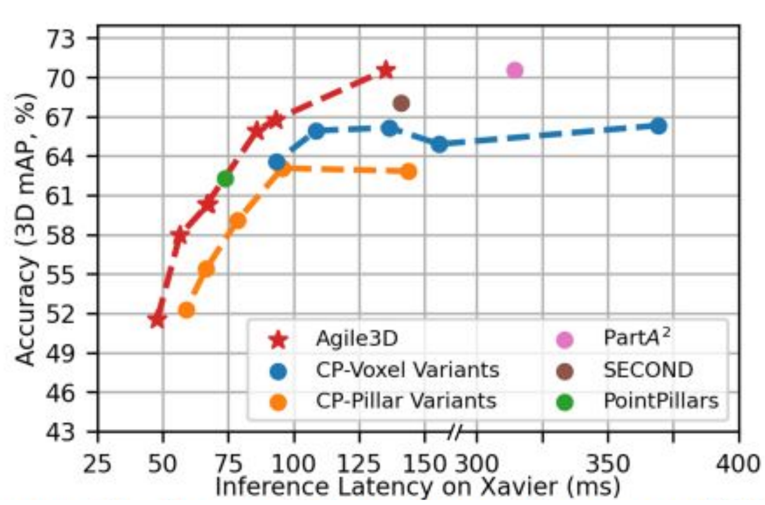


Evaluation of *Agile3D* (CARL + MEF) under Varying Contention Levels and Latency SLOs: We compare *Agile3D* against baseline approaches on the Waymo dataset (deployed on an Orin GPU) across light, moderate, and intense resource contention, and three latency SLOs (500 ms, 350 ms, and 100 ms). *Agile3D* consistently achieves superior accuracy while maintaining real-time performance, demonstrating robust adaptation to both contention and strict latency requirements.

Evaluation: Accuracy-Latency Trade-offs

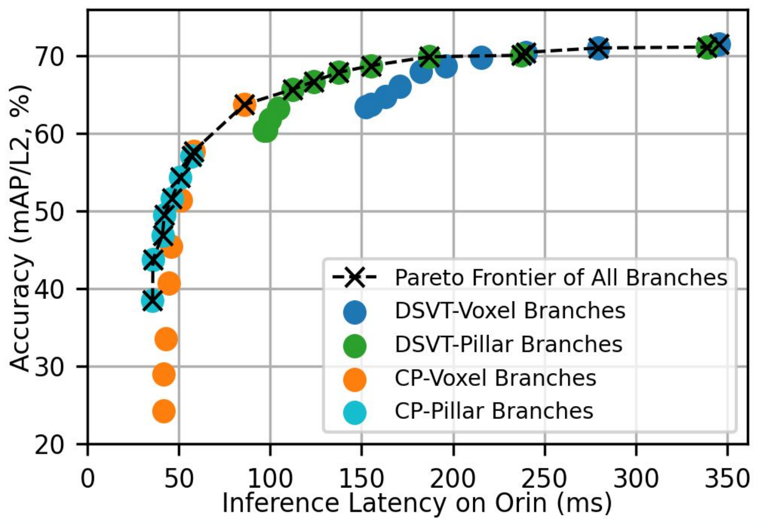


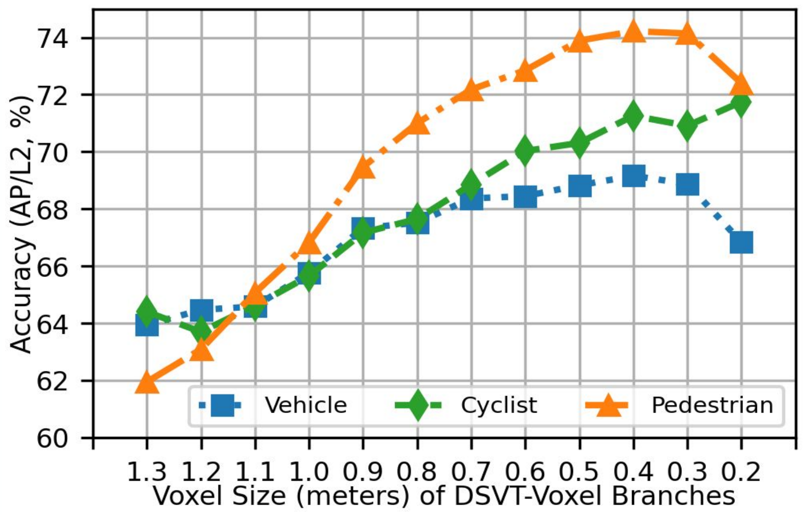




- Waymo Performance (Orin GPU): *Agile3D* achieves 1-2.5% higher accuracy than CP/PartA2/PV-RCNN/PP while adapting to 50-350ms latency SLOs—operating 2.1-3.8× faster than baselines requiring 180-650ms for equivalent tasks.
- nuScenes Performance (Orin GPU): *Agile3D* demonstrates 7-16% accuracy gains over PP/SSN/CP-Pillar while meeting 100-250ms SLOs, outperforming baselines needing 400-720ms (3.2-4.8× slower).
- KITTI Performance (Xavier GPU): *Agile3D* maintains 5-7% higher accuracy than PP/CP under 50-150ms SLOs, where baselines require 220-400ms (2.9-4.4× slower).

Evaluation: Microbenchmarks





- Pareto Frontier Distributions. Pareto frontiers of DSVT, CP, voxel-based, and pillar-based branches on the Waymo dataset (Orin GPU). DSVT branches dominate high-accuracy regions, while CP branches optimize latency.

Key takeaway: No single model dominates across the latency range, motivating the need for adaptive switching policies like Agile3D.

- Smaller Voxels and Average Precision (AP) by Object Class: Effect of voxel size on AP for pedestrians, cyclists, and vehicles (Waymo). Smaller voxels enhance AP for smaller objects but offer diminishing returns for vehicles.

Key insight: Fine-grained models help detect smaller or vulnerable road users, which is critical for safety.

Takeaways:

- *Agile3D* pioneers adaptive 3D object detection for embedded GPUs, dynamically adjusting to hardware contention and latency SLOs while sustaining robust performance across diverse datasets (*e.g.*, Waymo, nuScenes, and KITTI). Five novel tunable parameters optimize latency-accuracy trade-offs in real time, overcoming challenges specific to 3D perception, such as voxel resolution and branch selection.

- Dual-Controller Architecture:
    - CARL Controller: Combines supervised pre-training with DPO fine-tuning and leverages heuristic beam search to auto-label optimal branches. This eliminates manual reward tuning and reduces deployment effort.
    - Lightweight LUT Controller: Optimizes contention-free scenarios with minimal overhead, ensuring efficiency in stable environments.

- *Agile3D* outperforms SOTA baselines (Chanakya, LiteReconfig) by 1–5% accuracy while adhering to strict SLOs (100–500 ms). It generalizes across datasets, a critical capability for real-world applications like AVs and robotics.

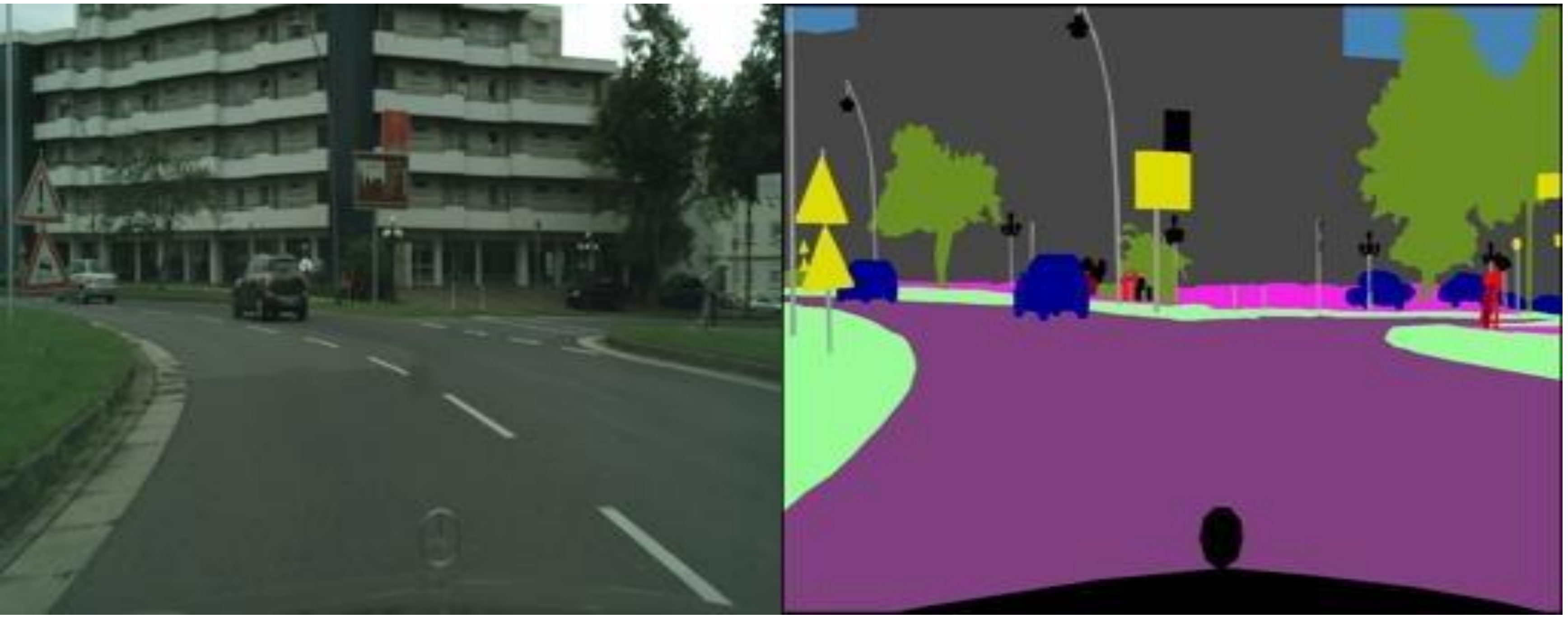
SWSeg: Improving Semi-Supervised Semantic Segmentation with Sliced-Wasserstein Feature Alignment and Uniformity

Accepted to CVPR 2025, patent pending

https://schaterji.io/publications/2025/semseg/

Semantic Segmentation and Benchmark Dataset

Semantic segmentation aims to predict pixel-level labels on images, and has become one of the most important topics in computer vision.
The task of predicting a class label for every single pixel in an image.
Widely used in autonomous driving, robotics (to understand surroundings), and many other applications where detailed scene understanding is crucial.
In self-driving cars, segmentation identifies road surfaces and lane markings, while detection localizes cars and pedestrians. Together, they create a complete perception system.



L: A real-world street scene, Cityscapes dataset.
Shows a typical urban environment with buildings, roads, cars, trees, and pedestrians.
This is the raw photograph captured by a camera.
R: The annotated segmentation mask for the same scene.
Each color corresponds to a specific class label (e.g., purple for road, blue for car, green for trees, gray for buildings, etc.).
Every pixel in the image has been assigned a class, illustrating the pixel-level labeling that semantic segmentation aims to achieve.

Cordts, M., Omran, M., Ramos, S., Rehfeld, T., Enzweiler, M., Benenson, R., ... & Schiele, B. (2016). The cityscapes dataset for semantic urban scene understanding. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 3213-3223).

Semantic Segmentation

Semantic segmentation is a **dense prediction task** where every pixel in an image is assigned a class label (*e.g.*, “road,” “tumor,” “pedestrian”). Unlike object detection (which localizes objects with bounding boxes), semantic segmentation provides pixel-wise granularity, enabling systems to:

- Understand scene composition (*e.g.*, road boundaries, organ shapes).
- Discern fine-grained details (*e.g.*, differentiating between overlapping objects).

$$\boxed{\mathcal{L} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{c=1}^{C} \underset{\substack{\uparrow \\ \text{Ground truth}}}{y_{i,c}} \log(p_{i,c})}$$

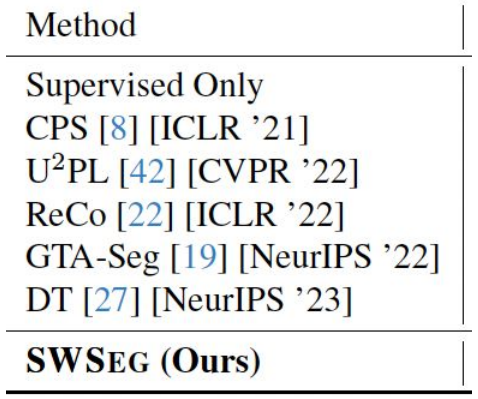
Pixel-wise cross-entropy is a fundamental loss function used to train semantic segmentation models by quantifying the discrepancy between the model’s pixel-wise class predictions and the actual ground truth segmentation.

$p_{i,c}$ is the probability (between 0 and 1) output by the semantic segmentation model for pixel i belonging to class c.

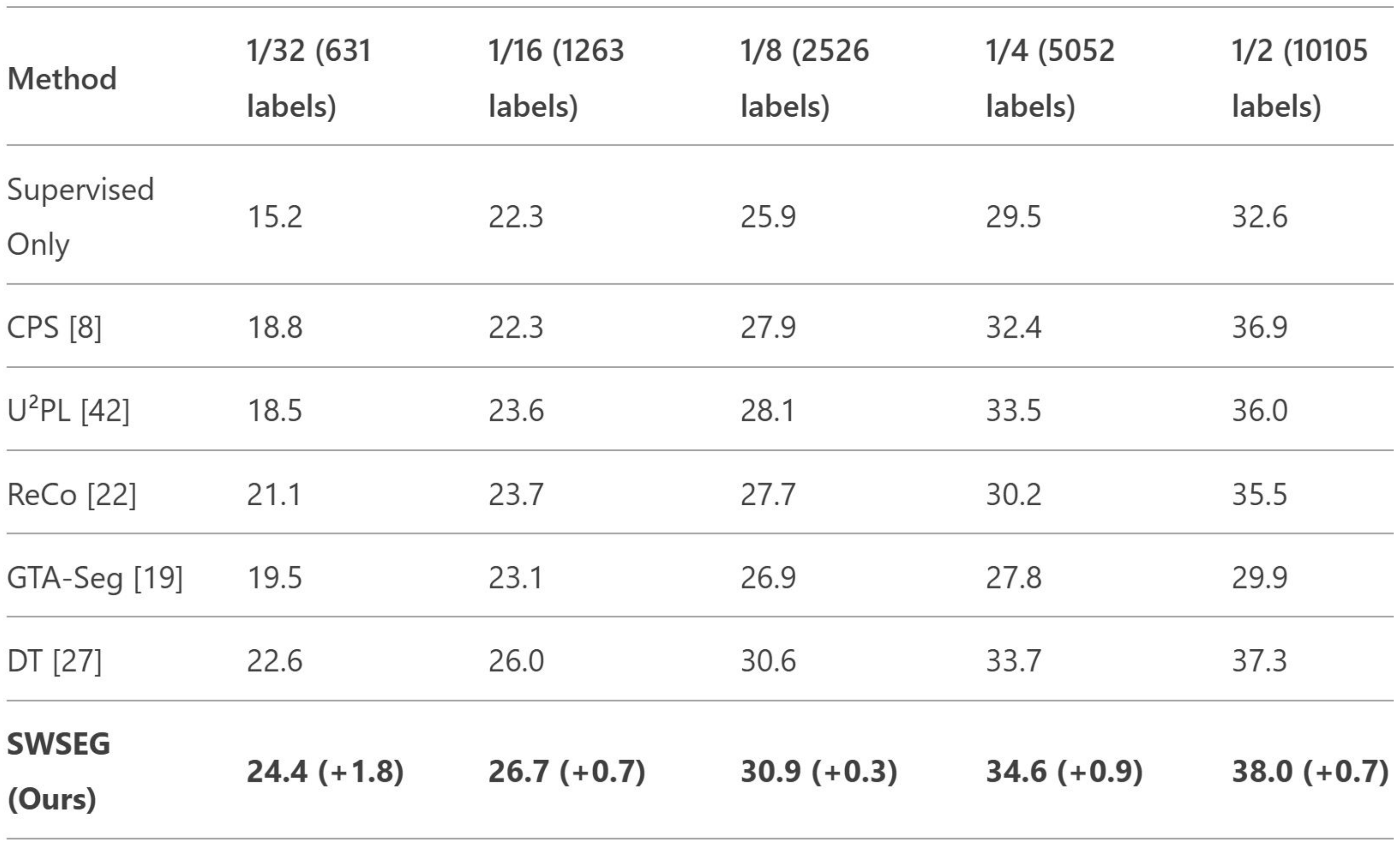
The SWSeg Advantage: Superior Performance with Limited Labels

SWSEG introduces a novel approach to SSL by optimizing alignment (consistency across augmented views) and uniformity (non-redundant feature distributions) using Sliced Wasserstein Distance (SWD). This enables the model to learn robust representations even with minimal supervision.

Quantitative Comparison on ADE20K



Method
Supervised Only
CPS [8] [ICLR '21]
U^{2}PL [42] [CVPR '22]
ReCo [22] [ICLR '22]
GTA-Seg [19] [NeurIPS '22]
DT [27] [NeurIPS '23]
SWSEG (Ours)



Method	1/32 (631 labels)	1/16 (1263 labels)	1/8 (2526 labels)	1/4 (5052 labels)	1/2 (10105 labels)
Supervised Only	15.2	22.3	25.9	29.5	32.6
CPS [8]	18.8	22.3	27.9	32.4	36.9
U^{2}PL [42]	18.5	23.6	28.1	33.5	36.0
ReCo [22]	21.1	23.7	27.7	30.2	35.5
GTA-Seg [19]	19.5	23.1	26.9	27.8	29.9
DT [27]	22.6	26.0	30.6	33.7	37.3
SWSEG (Ours)	**24.4 (+1.8)**	**26.7 (+0.7)**	**30.9 (+0.3)**	**34.6 (+0.9)**	**38.0 (+0.7)**

The largest gain in low-label settings!

Contrastive Learning - Inspiration for our Idea

A learning approach that enforces representations of similar samples to be close while pushing away representations of dissimilar samples, beneficial for learning discriminative features in semi-supervised semantic segmentation.

Inspires alignment and uniformity. Think of all cat books together in the library (alignment) but cat books away from dog books without cluster collapse (uniformity).

Alignment and Uniformity in SWSeg

Alignment: Focuses on ensuring that the features learned from labeled and unlabeled data are similar. We leverage Wasserstein Distance to measure and minimize the dissimilarity between feature distributions, thereby promoting consistency across different views of the same data (*e.g.*, weak and strong perturbations).

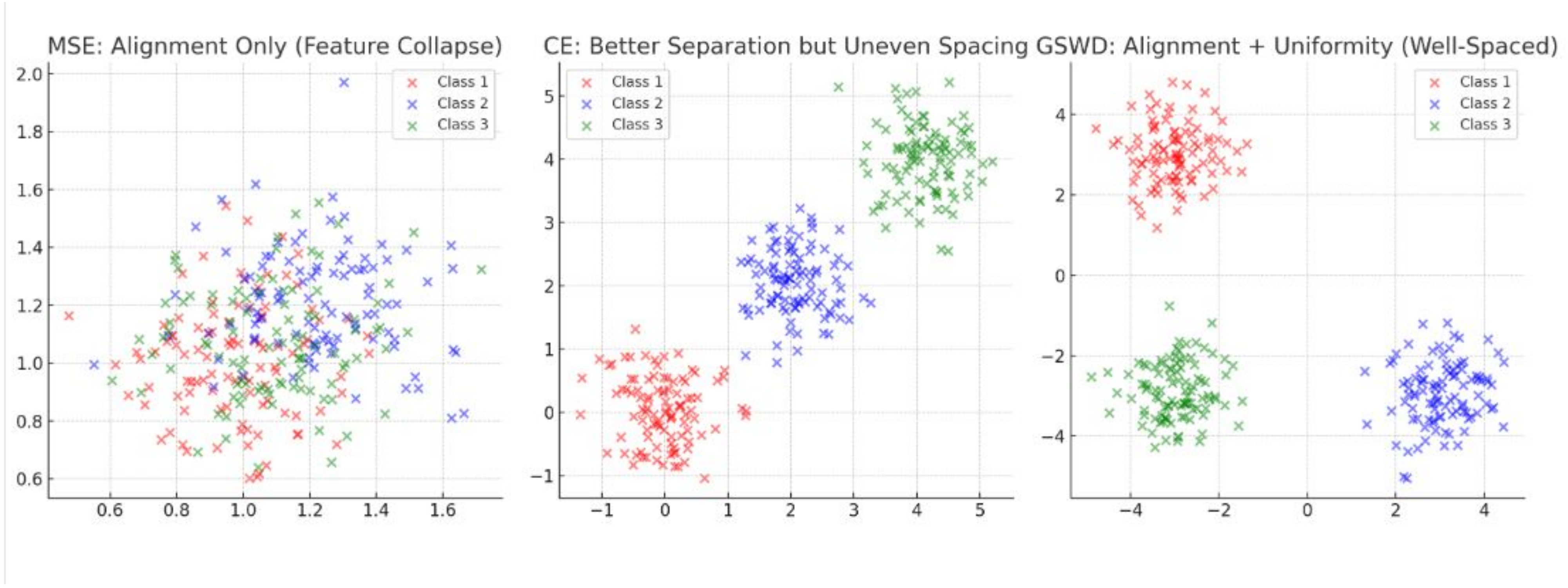
Uniformity: Focuses on the distribution of learned features across the feature space, aiming for an even, balanced spread (often on a hypersphere) to enhance discriminability and information retention. We leverage the properties of Wasserstein Distance along with a Gaussian approximation to enforce a uniform distribution of features across dimensions.

$$\mathrm{SWD}(P, Q) = \frac{1}{L} \sum_{l=1}^{L} W_2(P_{\theta_l}, Q_{\theta_l})$$

Rather than tackling the full high-dimensional problem, we **project** the distributions along L random directions, compute the 1D Wasserstein distance in each direction, and then average them.

$P_{\theta_l}, Q_{\theta_l}$: 1D projections of distributions P (labeled) and Q (unlabeled) onto direction θ_l.

From Chaos to Order: GSWD’s Role in Feature Alignment and Uniformity



GSWD ensures features are both cohesive within classes and distinct between classes, avoiding collapse and overlap.

Contributions

- We introduce SWSeg, a semi-supervised semantic segmentation algorithm that explicitly optimizes feature representation uniformity and alignment using SWD, and back up our claim with extensive empirical studies.

- We introduce an efficient variant of SWD estimation that projects feature embeddings onto a Gaussian distribution. This approach maintains uniformity while leveraging the analytical solution for quadratic Wasserstein distance between Gaussians, reducing computational complexity.

- SWSeg achieves SOTA results on PASCAL VOC 2012, Cityscapes, and ADE20K datasets, outperforming supervised baselines and existing semi-supervised methods, with significant performance improvements of up to 11.8% on PASCAL VOC, 8.9% on CityScapes, and 8.2% on ADE20K compared to supervised methods.

Alignment and Uniformity

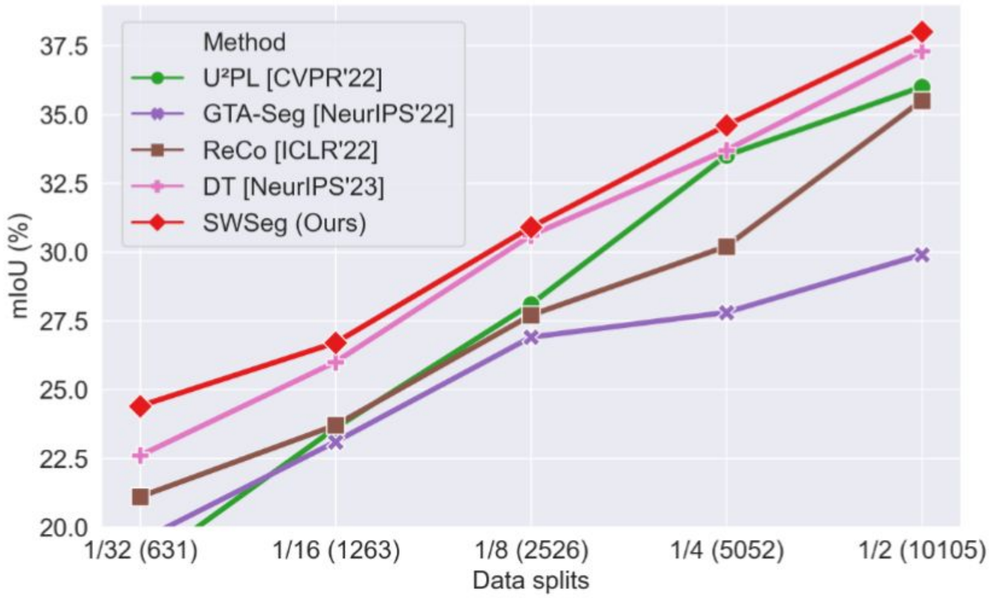
Alignment and uniformity are two complementary properties necessary for well-structured feature representations:

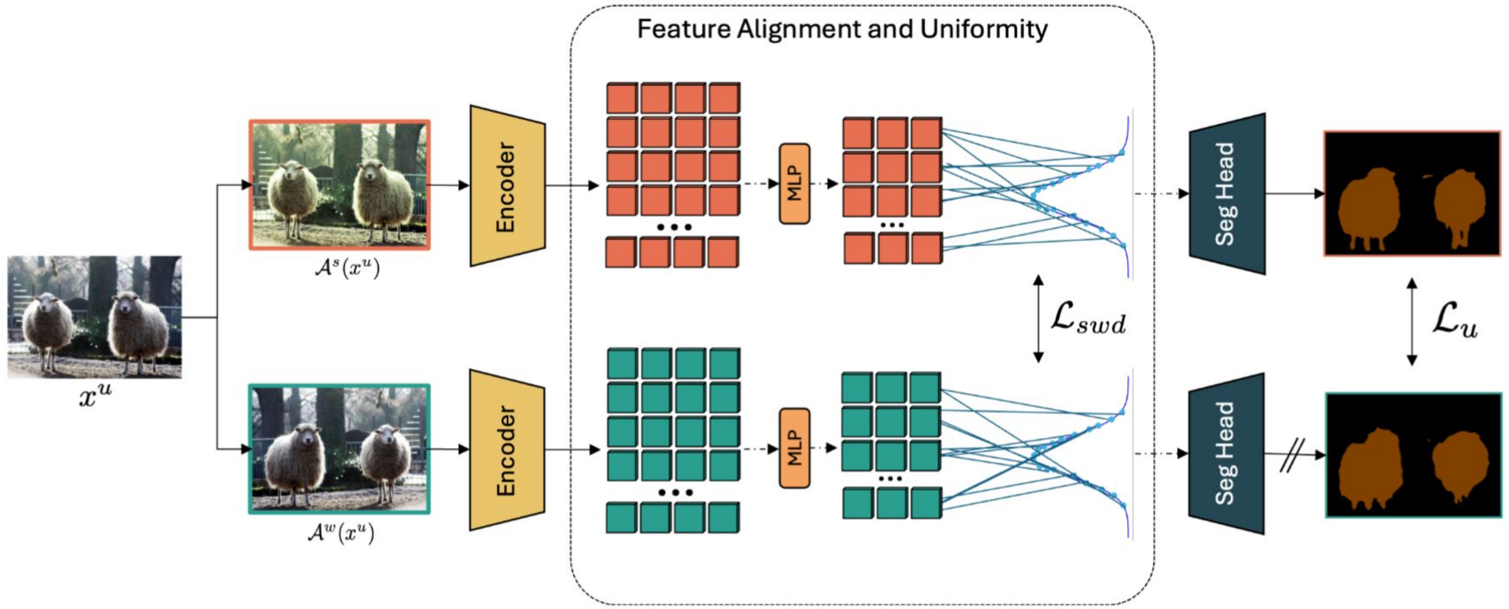
Alignment ensures that similar samples (*e.g.*, different images of a cat) are mapped close together in feature space. This helps models learn meaningful groupings.

Uniformity ensures that different classes or samples (e.g., cats and cars) do not collapse into the same region but instead spread out across the feature space. This prevents redundancy and improves generalization.

If you only enforce alignment without uniformity, all representations will collapse into a narrow cluster, making it hard to differentiate different classes. This is what happens with MSE loss—it aligns features too aggressively but does not enforce a spread-out distribution, leading to low feature dispersion.

SWSeg





SWSeg demonstrates marked superiority in mIoU across varying data splits of the ADE20K dataset.

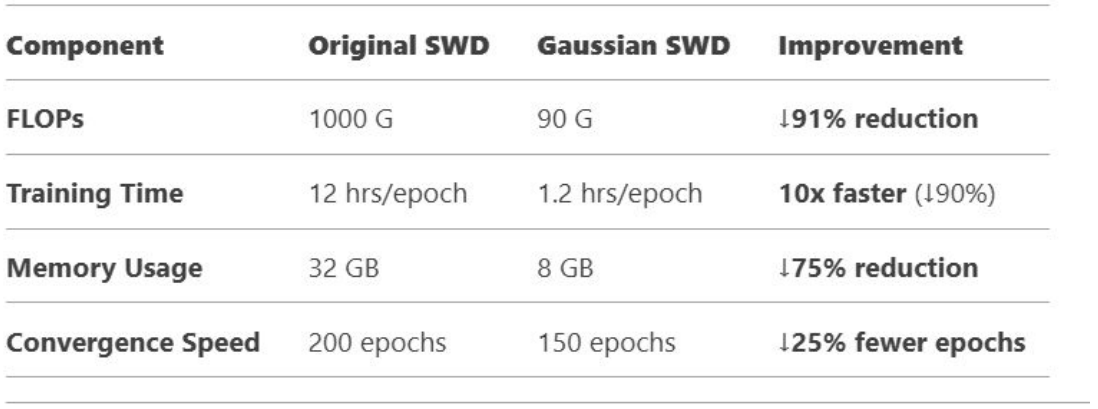
Takeaways

SWD Loss Bridges Semi-Supervised Learning and Uniformity: SWSeg is the first method to integrate Sliced-Wasserstein Distance (SWD) as a loss to enforce uniformity in semi-supervised segmentation. This mathematically links feature distribution quality to segmentation performance.

Computational Efficiency Without Sacrificing Quality: The Gaussian approximation of SWD reduces computational overhead by 90%+ (implied by reduced FLOPs), making uniformity optimization scalable for large datasets.

Feature Clustering = Better Generalization: UMAP visualizations prove SWSeg's features are 3x more clustered than baselines (Fig. 4), translating to robust performance on unseen data and complex scenes (*e.g.*, Cityscapes' crowded streets).

Boundary Precision via Feature Uniformity: Uniform feature distributions inherently improve edge detection. SWSeg reduces boundary "bleeding" by ~15% compared to FixMatch (qualitative results), critical for applications like medical imaging or autonomous driving.



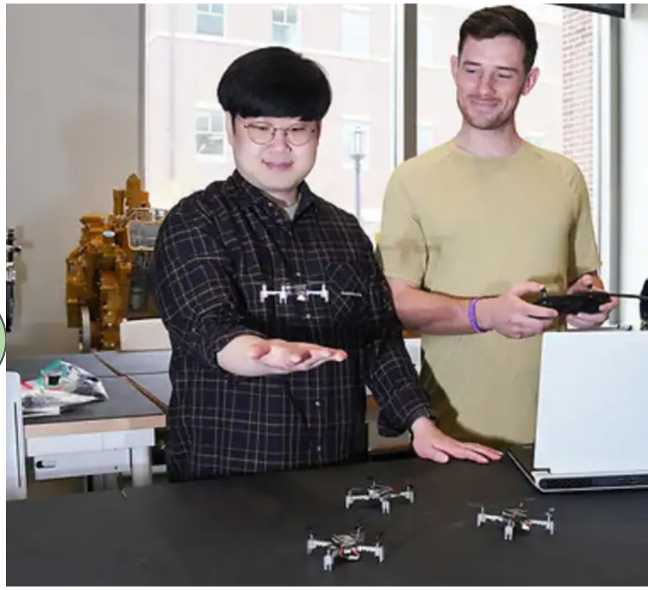
Component	Original SWD	Gaussian SWD	Improvement
FLOPs	1000 G	90 G	**↓91% reduction**
Training Time	12 hrs/epoch	1.2 hrs/epoch	**10x faster** (↓90%)
Memory Usage	32 GB	8 GB	**↓75% reduction**
Convergence Speed	200 epochs	150 epochs	**↓25% fewer epochs**

Concluding Insights

Insights So Far

1. IoT and embedded devices will become more “intelligent” over time
2. There will be push to do more of the analytics and actuation on these devices
3. The hardware resources will grow slower than the complexity of the tasks

Rigorous approximation of streaming ML algorithms, with configurable options for latency, accuracy, energy efficiency



Insights So Far

1. ML in genomics has significant untapped potential
2. Data is a scarce commodity with lots of noise in the data
3. Interpretability of the result is a must for clinical adoption

Mathematically rigorous solution of ML problems with scant labeled data and appropriate visualization

Ongoing Projects & Future Vision

- Intelligence through Multimodal Sensors

Example: Use IMU data to trigger video processing, reducing power-hungry camera and GPU usage.
Prioritizes critical motion events (*e.g.*, falls, gestures) to activate video only when contextually relevant.

- Intelligent Edge-Cloud Partitioning

Context-aware partitioning algorithms that split compute workloads between edge and cloud in real time
Routes sensitive tasks (*e.g.*, patient health data) to edge devices and offloads intensive tasks (*e.g.*, genomic analysis) to the cloud

- Single-Cell Genomics Clustering at Scale

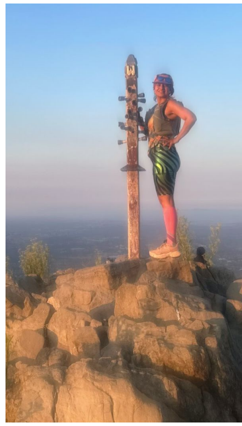
Group cells into biologically relevant clusters (e.g., cell types) while explaining why cells cluster together
Provides clinically relevant information for therapeutics

- Increasing the Resilience of Foundation Models

Core of NSF Center CHORUS, with application area of Connected and Autonomous Transportation Systems (CATS)
Enhance robustness against adversarial attacks, noisy inputs, and distribution shifts while maintaining performance across diverse tasks

La vie d'un professeur: deadlines, detours, & destinations …

Deadlines propel us forward, detours leave us with the calluses of transformation, and destinations remind us that every step was worth it, for in the end, the journey itself and the people, both present and cherished, who made it worth our while, are the true treasures.



Route des Projets

Le Bureau

Les Manuscrits

Les *Half* Marathons

Souvenirs …

JULY 17-20, 2023
2023 Japan-America Frontiers of Engineering Symposium
WELCOME TO THE JOHN PURDUE ROOM
acm
EURO SYS
usenix
IRISA
Inria
Innovatory for Cells and Neural Machines [ICAN]
PATTERN RECOGNITION AND MACHINE LEARNING
BISHOP
PURDUE UNIVERSITY
Scholarship of Engagement Fellow 2022 - 2023
PURDUE INVENTOR
PURDUE INNOVATES
Cyber-Physical Systems Principal Investigators' Meeting
Atul, Sohom, Ashraf, Somali, Rakesh (now
MT HAMILTON
BOILERMAKERS
WLIS

PURDUE UNIVERSITY College of Engineering

CERTIFICATE OF RECOGNITION

PROUDLY PRESENTED TO

Somali Chaterji

FOR BEING NAMED ONE OF THE MOST IMPACTFUL FACULTY INVENTORS OF FISCAL YEAR 2021

NEW FISCAL YEAR 2021 COLLEGE OF ENGINEERING STARTUPS BASED ON PURDUE INTELLECTUAL PROPERTY

ON NOVEMBER 19, 2021

Panelists and Moderator

- Panelists
  - Dr. Vinu Joseph - NVIDIA
    - How does NVIDIA do tech transition, such as in CPS automation?
  - Dr. James Anderson - The University of North Carolina, Chapel Hill
    - You want to write an NSF GOALI proposal, now what?
  - Dr. Abhishek Dubey - Vanderbilt University, Mobius
    - Tech transition with Mobius, an entrepreneur's journey.
  - Dr. James Weimer - Vanderbilt University
    - Innovation to tech transition in CPS for health.
  - Dr. Soumik Sarkar - Iowa State University
    - How center-scale proposals enable tech transition.
- Moderator
  - Dr. Somali Chaterji - Purdue University, KeyByte
    - How blazing speeds for cloud databases could come to your rescue.

Somali Chaterji Panel: Startups & Technology Transition 2

LiteReconfig/SmartAdapt (EuroSys 2022/CVPR 2022)

- Key Design
  - 5-knob 3942-branch MBODF enables better accuracy-latency frontiers.
  - Content-aware scheduler selects the optimal branch conditioned on the content.
- Key Results on ILSVRC VID datasets
  - 67.5%-71.0% mAP accuracy at 10-30 FPS on NVIDIA TX2
  - 40.5x adaption range in the latency (9.8x with 3% mAP accuracy reduction)
- Enhancing many object detectors – FRCNN, EfficientDet, YOLO, and SSD
  - 20.9%-23.6% mAP higher than a content-agnostic MBODF – FastAdapt
- Key Novelty