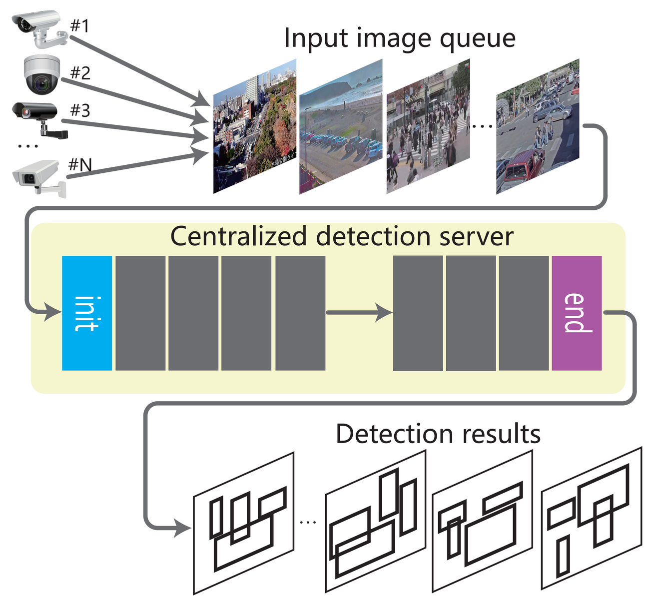
# Efficiency-preserving Scene-adaptive Object Detection

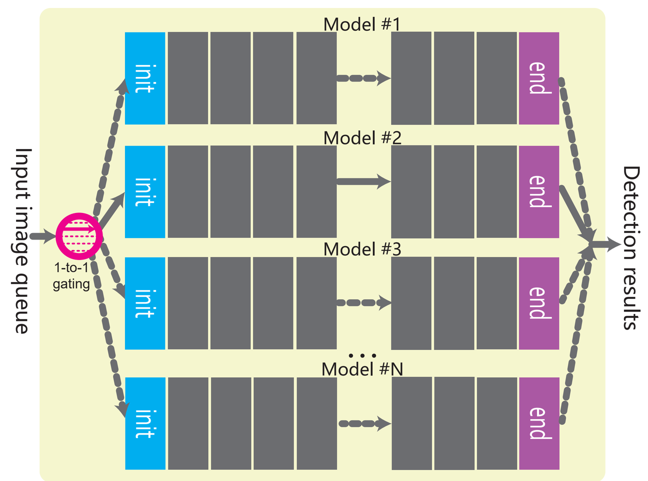
Zekun Zhang[1], Vu Quang Truong[2], Minh Hoai[3]

[1] Stony Brook University, [2] VinAI Research, [3] The University of Adelaide

## Problem Definition and Contribution

**Goal:** Presenting a framework that enables an object detector to self-enhance its accuracy while preserving its efficiency.




**Problem formulation:** Given a base detector $\mathcal{M}$, the core of an inference engine processing multiple camera streams, we aim to create an enhanced model $\mathcal{M}^*$ to replace $\mathcal{M}$ without increasing the overall parameter count by using separate models for each scene, which has:

- improved precision
- consistent latency and throughput
- memory efficiency
- self-supervised learning

## Intuition

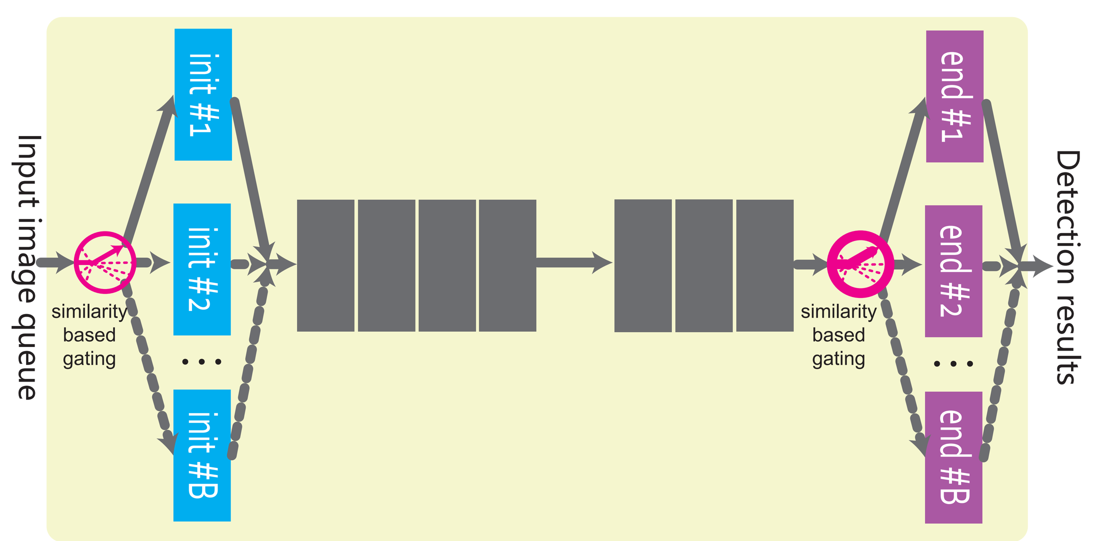
**Assumption:** In diverse scenes, the key differences are object sizes, camera perspectives, and lighting. Therefore, layers closest to the input or output are best for scene-specific adaptation. Early feature extractor layers adjust to geometric and lighting changes, while final head layers finalize the detector's decisions. Intermediate layers, focused on abstract visual representation, vary less across scenes.

**Main Idea:** Our method creates MoE architectures at the initial and terminal layers of the base model by duplicating them $B$ times. Each duplicated version represents an expert. We then propose a way to assign each video to each expert.

**Parameter count of the enhanced MoE model:** Let $|\mathcal{M}|$ be the parameter count of the original model, $\alpha$ the parameter proportion of scene-specific modules, and each of the scene-specific modules is duplicated for $B$ times. $B$ can be smaller than the number of scenes $N$, as a single module can still adapt to multiple similar video streams adequately. The parameter count of the enhanced MoE model is: $|\mathcal{M}^*| = (1+\alpha(B-1))|\mathcal{M}|$, which is significantly smaller than $B|\mathcal{M}|$.

## Method

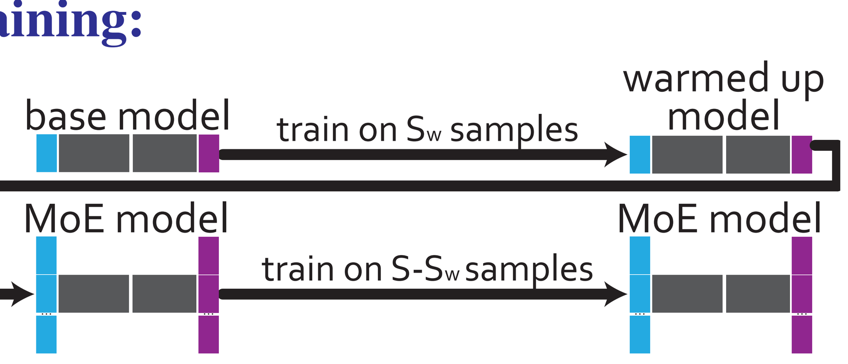
**Enhanced architecture:**




- We duplicate the initial and terminal layers. This method can be applied to various object detection architectures.
- Each branching junction utilizes a gating module determining the processing path for each input image based on its *scene ID*. We apply consistent gating rules across all gating modules.
- **The computational cost for processing each image remains the same as the base model.**

**Similarity-Based Gating:** Suppose we have to adapt the model to $N$ scenes. When $1 < B < N$, efficient scene-to-branch assignment is necessary. We take $M$ images from each scene and use a feature extractor $\mathcal{F}$ to extract their features. $B$-means clustering is applied to divide all the images into $B$ clusters. Major voting is utilized to determine the branch of each scene based on the cluster of the majority of its images.

**Two-stage training:**



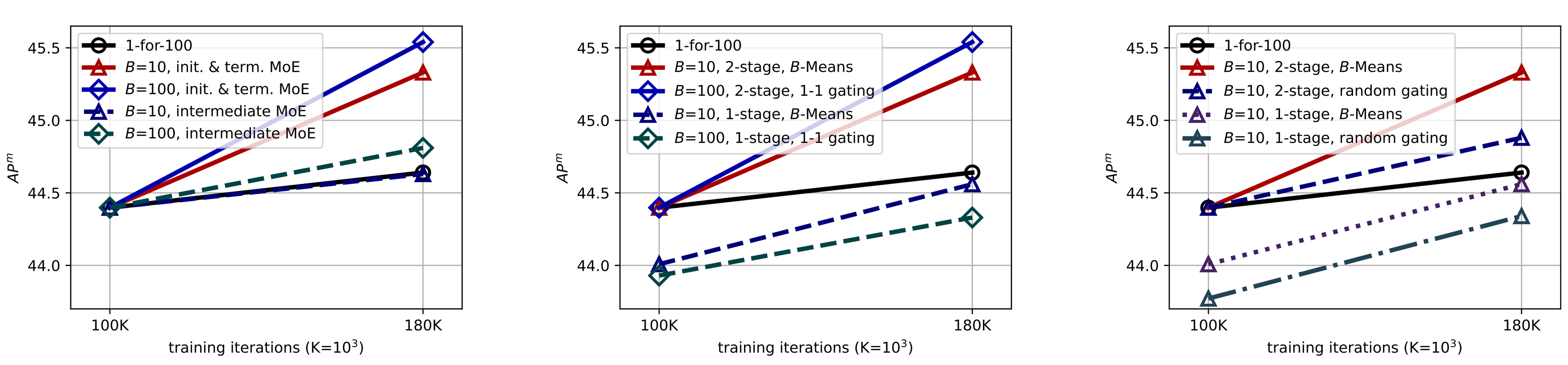

**Self-supervised learning:** We use the detection results from upscaling images of the base model as the pseudo-labels for training the enhanced model.

## Experiments & Results

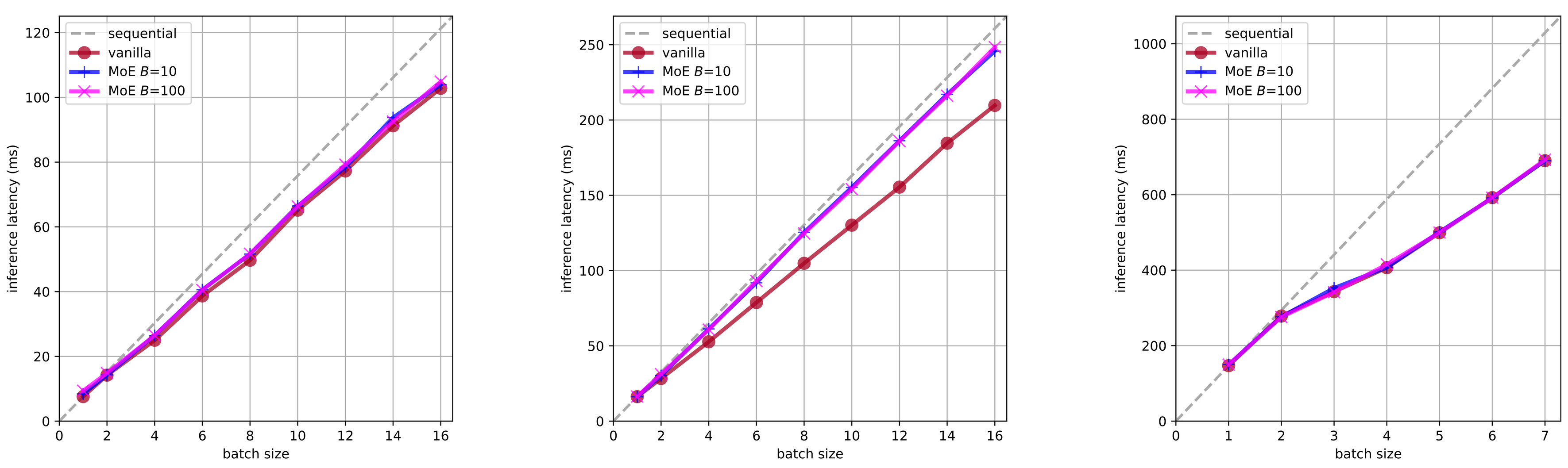
### Comparison with other adaptation methods on Scenes100

| Method | Pseudo label | Training samples ↓ | Deployment model size ↓ | GFLOPs ↓ | Relative latency ↓ | $AP^m$ ↑ |
|---|---|---|---|---|---|---|
| Base model (no adaptation) | Not applicable | | 230MB | 558 | 1.00 | 41.96 |
| ST [74] | DtTr | 8.00M | 230MB×100 | 558 | 1.00 | 43.35 |
| CT [89] | EnDtTr | 8.00M | 230MB×100 | 558 | 1.00 | 43.63 |
| MFM [89] | EnDtTr | 8.00M | 230MB×100 | 987 | 1.75 | 45.74 |
| GS [82] | EnDtTr | 8.00M | 231MB×100 | 1440 | 2.71 | 44.06 |
| LZU [81] | EnDtTr | 8.00M | 230MB×100 | 558 | 1.20 | 44.06 |
| LODS [45] | teacher | 0.10M | 230MB×100 | 558 | 1.00 | 42.98 |
| Proposed ($B$=10) | ×2 | 1.08M | 259MB | 558 | 1.01 | 50.27 |
| Proposed ($B$=100) | ×2 | 1.08M | 547MB | 558 | 1.00 | **50.39** |

### Ablation study on Faster-RCNN with ResNet-18 backbone




### Inference latency of different models at different batch sizes




a) YOLOv8s D-53 b) Faster-RCNN R-18 c) DINO-5sale R-50

### Comparison with the baseline of adapting the base model to all scenes with different detector architectures

| Base model | Method | Pseudo label | Training samples ↓ | Deployment model size ↓ | GFLOPs ↓ | Relative latency ↓ | $APD^m$ ↑ |
|---|---|---|---|---|---|---|---|
| Faster-RCNN with R-101 backbone | 1-for-100 | ×2 | 1.08M | 230MB | 558 | 1.00 | 0 |
| | 1-for-100 (no adapt) | Not applicable | | 230MB | 558 | 1.00 | −7.70 |
| | 1-for-1×100 | ×2 | 1.08M | 230MB×100 | 558 | 1.00 | 0.42 |
| | Proposed ($B$=10) | ×2 | 1.08M | 259MB | 558 | 1.01 | 0.61 |
| | Proposed ($B$=100) | ×2 | 1.08M | 547MB | 558 | 1.00 | **0.73** |
| Faster-RCNN with R-18 backbone | 1-for-100 | ×2 | 1.08M | 107MB | 310 | 0.54 | 0 |
| | 1-for-100 (no adapt) | Not applicable | | 107MB | 310 | 0.54 | −8.96 |
| | 1-for-1×100 | ×2 | 1.08M | 107MB×100 | 310 | 0.54 | 0.37 |
| | Proposed ($B$=10) | ×2 | 1.08M | 134MB | 310 | 0.54 | **0.90** |
| | Proposed ($B$=100) | ×2 | 1.08M | 397MB | 310 | 0.54 | 0.69 |
| YOLOv8s with D-53 backbone | 1-for-100 | ×2 | 0.95M | 43MB | 73 | 0.22 | 0 |
| | 1-for-100 (no adapt) | Not applicable | | 43MB | 73 | 0.22 | −1.63 |
| | 1-for-1×100 | ×2 | 0.95M | 43MB×100 | 73 | 0.22 | **0.98** |
| | Proposed ($B$=10) | ×2 | 0.95M | 55MB | 73 | 0.23 | 0.73 |
| | Proposed ($B$=100) | ×2 | 0.95M | 174MB | 73 | 0.24 | 0.75 |
| DINO-5scale with R-50 backbone | 1-for-100 | ×2 | 0.02M | 181MB | 1620 | 5.58 | 0 |
| | 1-for-100 (no adapt) | Not applicable | | 181MB | 1620 | 5.58 | −8.96 |
| | 1-for-1×100 | ×2 | 0.02M | 181MB×100 | 1620 | 5.58 | 1.34 |
| | Proposed ($B$=10) | ×2 | 0.02M | 198MB | 1620 | 5.71 | **2.45** |
| | Proposed ($B$=100) | ×2 | 0.02M | 373MB | 1620 | 5.85 | 1.95 |

### $AP^m$ and latency of $B = 10$ MoE models for different detector architectures at different input scale