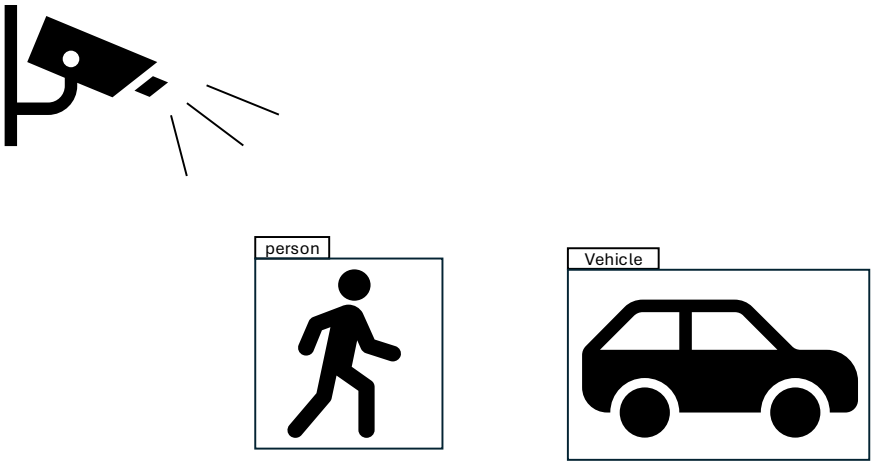
CHALMERS
UNIVERSITY OF TECHNOLOGY



Self-Supervised Fixed-Scene Adaptation for Object-Detection in Real-Time Surveillance: A Comparative Study of YOLO11 and RF-DETR

Master's thesis in Electrical Engineering

Jacob Justad

Department of Electrical Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY

Master's thesis 2026

Self-Supervised Fixed-Scene Adaptation for Object-Detection in Real-Time Surveillance: A Comparative Study of YOLO11 and RF-DETR

Jacob Justad

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Electrical Engineering
Chalmers University of Technology
2026

Self-Supervised Fixed-Scene Adaptation for Object-Detection in Real-Time Surveillance: A Comparative Study of YOLO11 and RF-DETR
Jacob Justad

Supervisor: Ass Prof Jennifer Alvén, Electrical engineering
Advisors: Martin Ljungqvist and Tiger Moberg, Axis Communications AB
Examiner: Ass Prof Jennifer Alvén, Electrical engineering

Master's Thesis 2026
Department of Electrical Engineering
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000 0

Gothenburg, Sweden 2026

Self-Supervised Fixed-Scene Adaptation for Object-Detection in Real-Time Surveillance: A Comparative Study of YOLO11 and RF-DETR
Jacob Justad
Department of Electrical Engineering and Engineering
Chalmers University of Technology

Abstract

This thesis investigates self-supervised fixed-scene adaptation for real-time object-detectors in an edge-computing surveillance context. While modern object-detectors achieve strong results on general-purpose benchmarks, deployment in static camera scenes introduces distinct challenges: domain shift to a specific viewpoint, limited availability of scene-specific labels, and stringent compute and memory budgets on-device. At the same time, the stationary background of surveillance footage provides exploitable structures, as do their temporal dependencies of video-frames. This study conducts a comparative analysis of two state-of-the-art object detection architectures: the Transformer-dominant RF-DETR and the convolutional neural network (CNN)-dominant YOLO11. The thesis employs the 100Scenes dataset to represent a broad range of surveillance environments. Experimental results demonstrate that RF-DETR consistently achieves higher accuracy, smoother convergence, and greater robustness than YOLO11, albeit with higher hardware demands. In contrast, YOLO11 variants (with a frozen backbone) leverage the larger trainable capacity of the neck and head to enable high scene-specific adaptability. While this yields significant gains under quality labeling, it tends to increase sensitivity to imperfect pseudo-labels and the risk of overfitting. Furthermore, by systematically varying model scales, adaptation strategies and environmental conditions the experimental design yields more than 3400 distinct runs. First the work examines the extent to which smaller, specialized models can match the approach of substantially larger models. The experimental results show that a small specialised model can compete with larger general models. Secondly, the study evaluated a proposed on-device self-supervised labeling strategy that integrates SAHI with a bidirectional implementation of ByteTrack. The proposed self-supervised labeling strategy provided reliable performance gains across all architectures and configurations, by recovering hard negatives, more specifically small, occluded and low confidence instances. Thirdly, the study investigated background-context fusion (BF). It proved to be consistently improving the performance in general for RF-DETR, while it proved inconsistent for YOLO11 and failed to increase robustness against seasonality, suggesting it induced background-dependent overfitting. Finally, the study shows that all models being trained on a summer scene exhibit a decrease in relative performance compared with the non-adapted models during a seasonal domain shift to a winter scene.

Keywords: Computer Vision, YOLO11, DETR, RF-DETR, Object detection, LW-DETR, YOLO, Self-Supervised Learning

Acknowledgements

I would like to express my sincere gratitude to Axis Communications for providing the opportunity to conduct this thesis and for giving me access to the resources to complete this work.

A special thanks goes to my supervisors at Axis, Martin Ljungqvist and Tiger Moberg.
I am grateful to my academic supervisor, Jennifer Alven, for your support, academic direction, and for ensuring the rigor of this research.

Jacob Justad, Gothenburg, 2026-02-08

Contents

1

Introduction

1.1 Background

The field of real-time object detection has evolved greatly in recent years, mainly due to rapid breakthroughs in deep learning and image classification, with the Convolutional Neural Network (CNN) [1] serving as the main keystone, making traditional feature extraction methods, like Scale-Invariant Feature-Transform (SIFT) and Histogram of Oriented Gradients (HOG), obsolete as standalone techniques. Furthermore, traditional machine learning models for classification, such as Support Vector Machines (SVMs), have been replaced by the fully-connected layers of perceptrons within the CNN architecture.

While these new CNN's architectures provide a powerful new foundation for classification, they do not inherently solve the full problem of object detection. This task is more complex, requiring the model to perform both localisation (finding where an object is) and classification (identifying what that object is) for potentially many objects at once. The challenge, especially for real-time applications, is to create architectures that could perform both of these tasks efficiently and accurately.

The release of the YOLO model in 2015 [2] transformed object detection into a one-stage problem while also providing the capability to solve the latency bottleneck (processing 155 FPS), allowing real-time surveillance and the development of intelligence directly in the camera. The YOLO lineage has since matured[3], [4], [5], [6], [7], [8], [9], [10], [11], [12] and has become the industry standard for balancing speed and accuracy on constrained hardware.

Today's industry faces a paradigm shift originating from Natural Language Processing. Following the widespread adoption of transformers [13] in powering Large Language Models (LLMs), the architecture was also adapted for image analysis. Vision

Transformers (ViTs) challenged the CNN monopoly by treating image patches as sequences, demonstrating performance superior to state-of-the-art convolutional networks in classification tasks [14]. While the original Detection Transformer (DETR) introduced a revolutionary end-to-end pipeline for object-detection, its prohibitive computational cost precluded edge deployment [15]. Recent innovations have dismantled this barrier, evolving from Lightweight DETR [16] to the state-of-the-art Roboflow Detection Transformer (RF-DETR) [17]. Crucially, RF-DETR diverges from previous architectures by integrating a DINOv2 [18] backbone, effectively distilling the robust features of a self-supervised foundation model into a real-time framework. However, transformer architectures are still more hardware-demanding.

Hardware limitations are the main bottleneck in edge surveillance. Because superior model performance generally correlates with higher hardware requirements, scaling advanced analytics can quickly become cost-prohibitive. The vital business case, therefore, lies in optimisation. By validating that efficient, lightweight models can rival the accuracy of computationally intensive ones, companies can deploy premium analytics on cost-effective hardware—cutting operational costs while maintaining high performance.

The most recent progress in object detection is typically benchmarked on COCO [19], a general-purpose dataset that has driven impressive gains but does not fully reflect fixed-camera surveillance. In a static scene, the camera viewpoint is constant and the background is repeatedly observed. This makes deployment different in two important ways. First, detectors trained on broad datasets often degrade under domain shift to a specific scene [20]. Nevertheless, collecting and annotating scene-specific data at scale is costly and sometimes sensitive. This motivates self-supervised scene adaptation from unlabeled footage to specialize pre-trained detectors without manual labels [21]. This is a necessity for commercial use. Secondly, the static background is not just irrelevant context, it can be exploited, as shown by Zhang et al. [21] for an older model architecture called Faster R-CNN [22]. This motivates investigating whether background extraction and feature fusion can systematically increase performance in the currents state-of-art model architectures.

1.2 Aim

The primary aim of this thesis is to evaluate the performance and trade-offs of self-supervised fixed scene adaptation for real-time object detectors in edge-computing environments.

The thesis focuses on evaluating two state-of-the-art real-time object-detector architectures currently dominating the field: The CNN-dominant architecture represented by YOLO11 and the transformer-dominant architecture represented by RF-DETR. I am examining how effectively the models adapt to a specific scene in order to better

understand the influence of architectural choices.

In an edge-computing environment, hardware constraints are inevitable. Thus, the adaptability of the models across different sizes will be explored further by experimenting using different model sizes. Further, manually labeling potentially thousands of specific scenes is infeasible, making self-supervised learning a cost-effective choice. This thesis investigates how far a potential on-device compatible pseudo-labeling strategy can perform in relation to existing labeling methods. Taking advantage of the stationary nature of surveillance cameras has been achieved through background extraction and feature fusion in previous architectures, boosting performance [21].Thus, I further aim to examine whether an adaptation of Zhang et al.'s method for exploiting static backgrounds[21] can be used to enhance the performance of YOLO11 and RF-DEFR.

In real-world deployments, environmental and operational conditions vary over time (e.g., across seasons and weather regimes), rendering robustness and generalisation properties critical. Consequently, this thesis additionally investigates the extent of performance degradation under such distributional shifts to more rigorously characterize the risks associated with model specialisation, such as overfitting.

1.3 Research Questions

Based on these objectives, the thesis addresses the following specific research questions:

1. RQ1 : How do Yolo11 (CNN) and RF-DETR (Transformer) compare in adaptation capability within static scenes?

2. RQ2 : To what extent can self-supervised scene adaptation enable smaller models compared to larger ones?

3. RQ3 : Can a potential on-device strategy for self-supervised learning achieve performance in parity with more resource-heavy methods?

4. RQ4 : Does the integration of background extraction and feature fusion in a fixed-camera environment provide a performance improvement for the chosen models?

5. RQ5 : How does the accuracy of the adapted model under seasonal shifts compare to the performance of the non-adapted base model and its trained-domain performance?

1.4 Limitations

The scenes have only been evaluated once per model configuration. Thus, the same seed is being used. There is no statistical analysis per scene more than qualitative analysis and once seasonality is being investigated.

Hardware and latency validation is a primary limitation of this study, as I rely on reported latency figures from existing literature rather than direct on-device benchmarking. Crucially, no independent latency measurements were conducted in this study. However, obtaining these metrics is essential to accurately evaluate the real-world computational trade-offs. The comparison between the models is further complicated by inconsistencies in prior works, such as the mixed use of FP16 for latency measurement versus FP32 for accuracy assessment. I am using FP32 for both, however for RF-DETR mixed-precision is being utilized in training.

I conducted preliminary latency measurements for the models used during training without applying any optimizations. However, the observed latencies deviated from the theoretically reported values in the existing literature. To enable a rigorous and fair comparison, additional work is required in terms of model optimization and exporting the models to the appropriate formats.

The experimental scope was restricted to a fixed input resolution of 640×640, however, generalisation of these findings to other resolutions remains unproven.

Additionally, due to time and resource limitations, I did not include most recent state-of-the-art architectures such as D-Fine [23], but instead focused on more established architectures.

2

Preliminaries

2.1 Traditional computer vision-based object detection

Traditional computer vision (CV) approaches to object detection are defined by a multi-stage, "handcrafted" pipeline: Preprocessing → Feature Extraction → Classification. Unlike modern approaches that learn features directly from data, traditional methods rely on manual engineering to transform raw pixel data into compact representations robust to variations in lighting, scale, and rotation. A prominent example of this era is the Scale-Invariant Feature Transform (SIFT) [24] which made a huge impact. SIFT was designed to match objects by identifying stable interest points ("blobs") and computing descriptors based on gradient orientations in the keypoint's neighborhood. Histogram of Gradients (HOG) [25] also showed promise in human detections. While effective for matching, using these descriptors for detection required a secondary step. Typically, descriptors were aggregated or analyzed using a "sliding window" approach, where a classifier scanned the image to identify object presence. Support Vector Machines (SVMs) became the state-of-the-art classifier for this task. Using the "kernel trick," SVMs find an optimal, maximum-margin hyperplane to separate object classes in the high-dimensional feature space created by descriptors like SIFT or HOG. However, the performance of these systems was fundamentally limited by the quality of the manually designed features.

2.2 Neural networks-based object detection

Neural networks represents a paradigm shift from handcrafted features to end-to-end learning. In this framework, the network learns to extract features for object detection directly from the training data

The base building block of the architectures of stacked layers known as deep neural

networks is the multi-layer perceptrons. Artificial neurons are arranged in layers, where each unit computes a weighted sum of its inputs. Crucially, this sum is passed through a non-linear activation function (e.g., ReLU or Sigmoid).Without these non-linearities, a stack of neural layers, no matter how deep, would mathematically collapse into a single linear transformation, rendering the network incapable of modeling complex decision boundaries. [26], [27], [28].

Training these networks is treated as an optimisation problem by minimizing a loss function that quantifies the error between the network's predictions and the ground truth. How these networks learn efficiently finally became a significant breakthrough, achieved through backpropagation. [29], [30]. Backpropagation applies the chain rule from calculus in two passes. In the forward pass, the input data is fed through the network, layer by layer, to compute the activations and the final output. This output is then used to calculate the value of the loss function. In the backward pass, the algorithm propagates the gradient of the loss backward through the network, starting from the output layer. At each layer, it efficiently computes the gradient of the loss with respect to that layer's parameters, crucially reusing the gradients computed for the layer "above" it. This dynamic programming approach avoids redundant calculations and makes training deep networks computationally feasible. The most widely adopted method for this is an optimisation technique called gradient descent. To manage the complexity of modern deep networks, adaptive optimizers are used. AdamW [31] is currently a standard choice. While its predecessor, Adam [32], adapted learning rates using moving averages of gradients, it handled regularisation suboptimal. AdamW decouples weight decay from the gradient update, applying it directly to the parameters. This modification significantly improves generalisation and training stability for deep neural networks.

2.3 Convolutional neural networks-based object detection

This section provides a brief overview of the convolutional network used in object detection and concludes by describing one of the main architectures of the YOLO11 model.

2.3.1 Classification and two-stage object detection

Hubel and Wiesel [33] research on a cat's primary visual cortex established that specific neurons are distinguished and possess a local receptive field, responding only to stimuli within a restricted region of the visual field. Furthermore, they demonstrated that many of these neurons are functionally selective, responding optimally to simple geometric structures such as oriented edges or bars. They could also show that it was a hierarchical processing model, wherein "simple cells" respond to these local features and feed this information to other, more "complex cells," which pool from more than one cell. These biological findings laid the groundwork for the

Neocognitron [34] and eventually the first practical CNN in 1998 by LeCun LeNet-5 [35]. This architecture combined convolutional layers with subsampling (pooling) layers and was trainable end-to-end using backpropagation. It extract local features through the stacking of layers, increasing the receptive-field, being the area in the original resolution a feature can derive from, which in a traditional CNN increases the deeper in the network it is. The weight-sharing mechanism significantly reduced the model's parameter count, enhancing parameter efficiency and generalisation, and its success in handwritten digit recognition demonstrated the viability of learned hierarchical features.

The modern era of Large Scale deep convolutional models was catalyzed by the ImageNet Large Scale Visual Recognition Challenge (ILSVRC). Krizhevsky, Sutskever, and Hinton's AlexNet [36] achieved a substantial reduction in top-5 classification error compared to traditional computer vision pipelines based on hand-designed descriptors such as SIFT and HOG.

This breakthrough pivoted object detection from sliding-window techniques to deep convolutional architectures. The foundational R-CNN [37] applied CNNs to region proposals generated by selective search, significantly improving accuracy but suffering from high latency due to redundant feature computations, introducing a two-stage object-detection model. Subsequent models optimized this pipeline: SPP-Net [38] and Fast R-CNN [39] introduced shared feature maps and Region of Interest (RoI) pooling, enabling end-to-end training for classification and regression. Faster R-CNN [22] eventually eliminated the external proposal bottleneck by introducing the Region Proposal Network (RPN), achieving a fully unified, near real-time two-stage detector.

2.3.2 YOLO - One-stage paradigm shift

In 2015, Redmon et al. proposed "You Only Look Once" (YOLO), a novel architecture that reframed object detection as a single regression problem rather than a classification task applied to region proposals [2]. Unlike two-stage methods (e.g., Faster R-CNN) that first generate candidate regions, YOLO processes the entire image in a single forward pass, enabling real-time inference.

The fundamental concept involves dividing the input image into an $S \times S$ grid. If an object's center falls within a grid cell, that cell is responsible for detecting it. Each cell simultaneously predicts B bounding boxes (coordinates x, y, w, h), a confidence score reflecting the intersection-over-union (IoU) with the ground truth, and conditional class probabilities. This idea is visualized in Figure 2.1.

Given that grid-cells may produce overlapping predictions for the same object, Non-Maximum Suppression (NMS) is applied during inference. NMS filters redundant

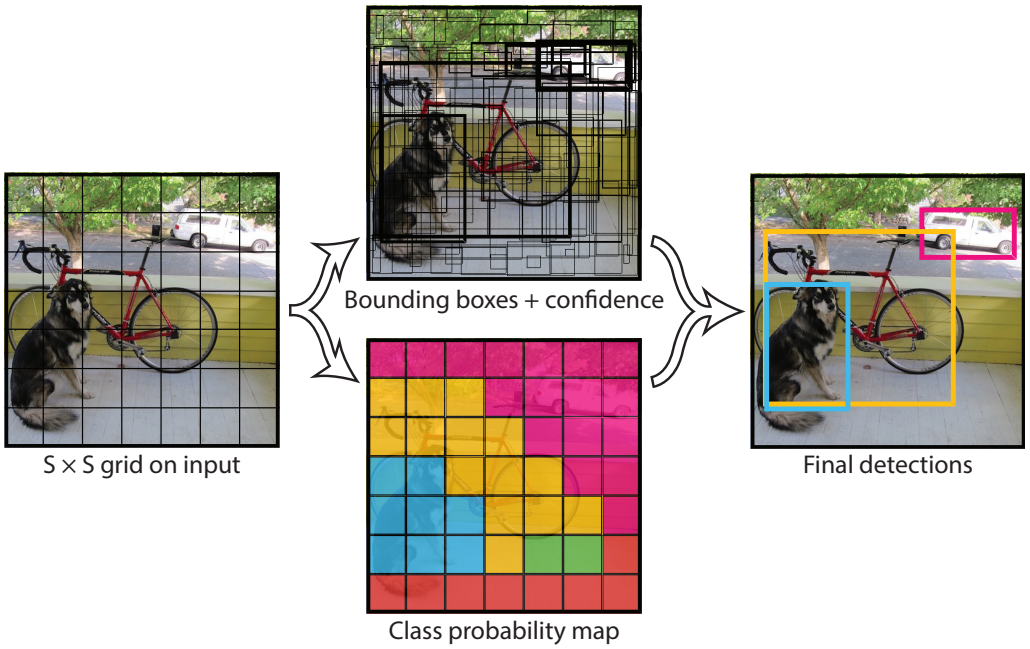


Figure 2.1: Object detection as a regression problem. YOLO divides the image into an S × S grid and for each grid cell predicts B bounding boxes, confidence for those boxes, and C class probabilities. These predictions are encoded as an $S \times S \times (B \cdot 5 + C)$ tensor, from the original paper [2]

detections by discarding boxes with low confidence and suppressing those that have a high overlap of IoU with the highest-scoring box for a given class.

Training is optimized via a multi-part Sum of Squared Errors (SSE) loss function that combines localisation and classification. This loss penalizes errors in the box center coordinates (x, y). A dimension loss is also added based on width and height (w, h); instead of squared errors, squared roots are used to align better with different object sizes, hoping to reflect that small deviations in large boxes matter less than in small boxes. Additionally, it regresses the objectiveness confidence toward the ground truth IoU, while a down-weighted "no-object" term suppresses false positives in background cells to prevent them from overwhelming the gradients. Finally, the optimisation includes a classification term that minimizes the error in class probabilities, but this is applied conditionally: it only penalizes classification errors if an object is present in the grid cell. This ensures the model effectively learns $P(Class|Object)$, ignoring class predictions for background cells

The YOLO-series has since then incrementally updated their model and improved both accuracy and latency. The YOLOv2 [40] most significant change was the adoption of anchor boxes, a concept borrowed from region-proposal-based networks like Faster R-CNN. The fully connected layers responsible for directly predicting bounding box coordinates in YOLOv1 were removed. Instead, the final convolutional layers were designed to predict offsets to a set of pre-defined prior boxes, or anchors.

Instead of hand-picking the anchor box priors, which can be suboptimal, YOLOv2 employed k-means clustering on the bounding box dimensions from the training dataset to automatically find a good set of priors. YOLOv3 introduced multi-scale detection using a Feature Pyramid Network (FPN)-like structure, extracting features at three different strides to detect objects of varying sizes [4], [41]. Subsequent versions like YOLOv4 and YOLOv5 introduced Cross-Stage-Partial (CSP) backbones and Path Aggregation Networks (PANet) for better feature extraction. They also formalized "Bag of Freebies" training techniques, replacing standard loss functions with CIoU and introducing Mosaic data augmentation [6]. Models continued evolving, and YOLOX in 2021 [42] introduced anchor-free predictions. Instead of predicting offsets to pre-defined anchor boxes, YOLOX treated detection as a per-pixel prediction problem. They also introduced a decoupled head, separating the classification and regression tasks into two parallel branches. To address the issue of assigning ground truth objects to the correct predictions for training, YOLOX incorporated an advanced dynamic label assignment strategy called SimOTA. Instead of relying on fixed IoU-based rules, SimOTA formulates label assignment as an optimal transport problem. YOLOv8 [9] adopted this and further made incremental improvements within the blocks of how the features are being passed. The latest iteration true to the convolutional architecture is the YOLO11 family of models, which represents the latest generation of YOLO-based CNN detectors as of 2024–2025, encompassing the accumulated advancements. Although both Yolo12,Yolo13, and Yolo26(soon to be released) are newer models, they differ from the more traditional YOLO and convolutional network architecture and are not as established as YOLO11.

YOLO11's architecture, like most detection systems, can be divided into 3 main modules: the backbone (feature extractor), the neck (feature fusion), and the head (predictors). YOLO11 remains true to its core by being a convolution-based architecture with grid-based cells at its center. This means that despite being anchor-free and more flexible, YOLO11 still divides the image feature maps into a dense grid, where each cell (spatial location in the feature map) is responsible for predicting object presence, bounding box offsets, and class probabilities. For YOLO11, 3 different spatial feature maps are used in the detectors head, which can be seen in the Figure 2.2. YOLO11 uses a configuration similar to CSP-based backbone [43] as its backbone for multi-scale feature extraction. However, it incorporates key modules such as C3k2 to improve feature representation (replacing the older C2f), SPPF (Spatial Pyramid Pooling Fast) to extract global semantic information, and the new C2PSA module, which uses pyramid slice attention to better identify objects in complex backgrounds and locate small objects. Following the backbone, the neck employs a PAN-FPN (Path Aggregation Network - Feature Pyramid Network) structure. This design allows for bidirectional (bottom-up and top-down) information flow, effectively fusing shallow spatial features with deep semantic features to enhance localisation accuracy. Finally, the head uses a decoupled architecture, handling classification and bounding box regression as separate tasks. A critical evolution in this architecture is the shift from static to dynamic label assignment. While the original YOLOv1 relied on a rigid geometric rule—assigning responsibility strictly to the single grid

cell containing the object's center YOLO11 employs Task-Aligned Learning (TAL). Instead of a binary assignment based on center location, TAL dynamically selects the top-k grid cells inside an object that maximize a high-order alignment metric:

$$t = s^{\alpha} \times u^{\beta} \tag{2.1}$$

where s is the predicted classification score and u is the Intersection over Union (IoU) with the ground truth. This ensures that the assigned positive samples maximize both classification confidence and localisation accuracy simultaneously. Unlike the hard labels in YOLOv1, TAL generates "soft" supervision targets, scaling the training signal based on the alignment quality t. The classification branch uses Binary Cross Entropy (BCE) Loss and an efficient depthwise convolution layer , while the regression branch combines Distribution Focal Loss (DFL) and Complete IoU (CIoU) loss to optimize bounding box accuracy and stability [12], [44], [45].

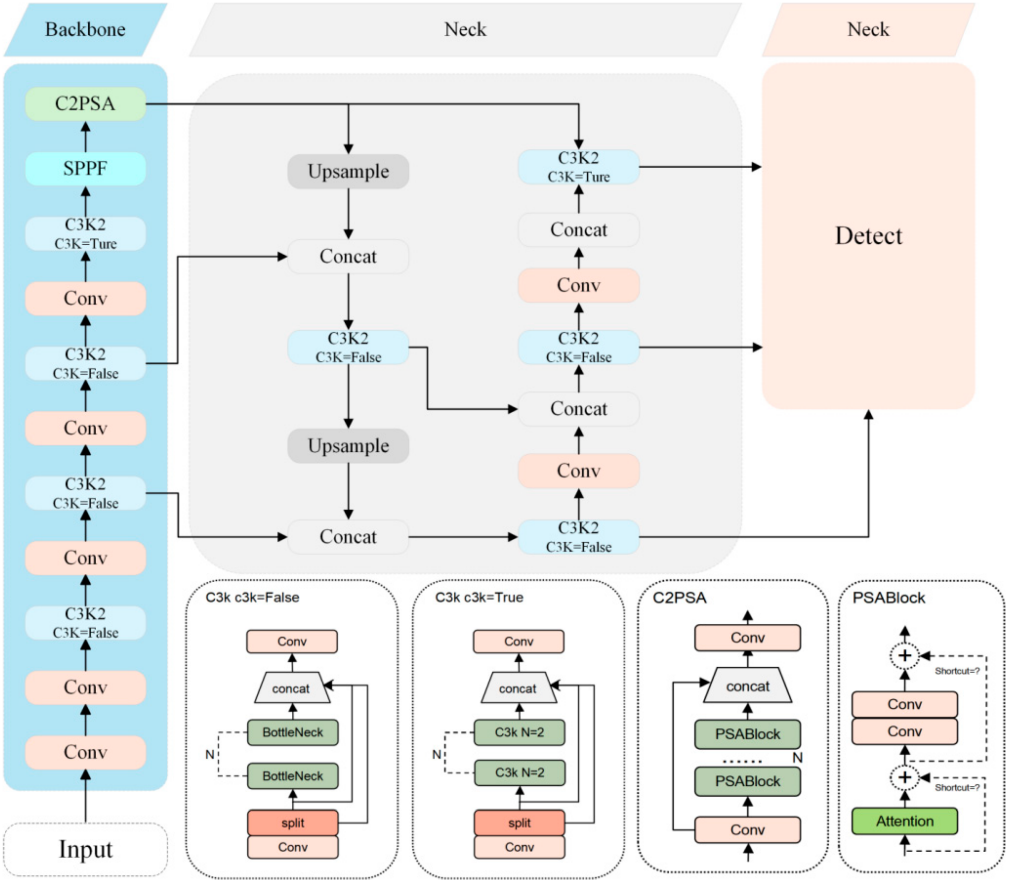


Figure 2.2: Detailed breakdown of the YOLO11 architecture. The input image is processed through a CSP-based backbone (blue) featuring SPPF and C2PSA modules, followed by feature fusion in the neck (grey) via upsampling and concatenation, leading to the final multi-scale detection output. From Fang et al. [44].

2.4 Transformer-based object detection

This section describes the attention mechanism and its application in computer vision. Finally, I introduce RoboFlow-DETR (RF-DETR), which is one of the main architectures examined in this study.

2.4.1 The attention mechanism and the Vision Transformer

The Transformer architecture marked a paradigm shift by demonstrating that recurrence was not a prerequisite for state-of-the-art sequence modeling [46]. The central hypothesis, "Attention Is All You Need," proposed dispensing with recurrence entirely and relying solely on attention mechanisms. The core component, self-attention (or intra-attention), allows the model to compute representations for each position in a sequence by attending to all other positions within that same sequence. To compensate for the loss of sequential order information, the model ingests positional encoding along with the input embeddings. This architecture is massively parallelized and has been one of the foundations of the progress we are seeing today in artificial intelligence.

In 2021, Dosovitkiy et al. introduced the Vision Transformer (ViT), successfully applying this architecture to computer vision [14]. ViT processes an image by dividing it into fixed-size patches (e.g., 16×16 in pixels), flattening them into linear embeddings, and treating them as a sequence of "words." A learnable "classification token" is prepended to the sequence to aggregate global information for the final prediction. This approach achieved excellent results compared to state-of-the-art CNNs while being highly computationally efficient to train.

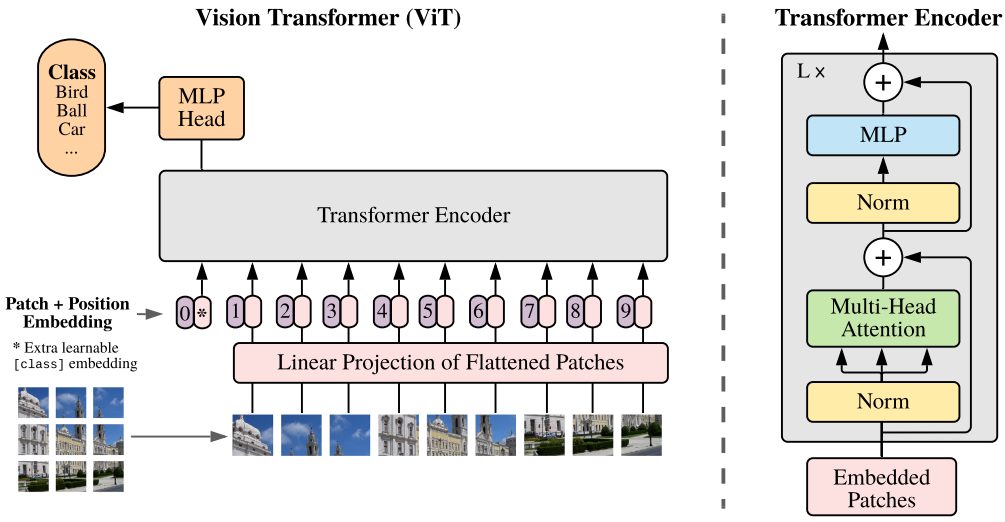


Figure 2.3: the image is split into fixed-size patches, each patch is converted into a vector, a positional embedding is added, and the resulting sequence of vectors is passed into a standard Transformer encoder. In order to perform classification, we use the standard approach of adding an extra learnable “classification token” to the sequence. The figure is retrieved from the orginal paper [14].

2.4.2 RoboFlow-DETR (RF-DETR)

The RF-DETR [47] represents the current state-of-the-art in object detection, culminating in a lineage that began with DETR [15]. DETR introduced a paradigm shift by utilizing set-based prediction via bipartite matching to eliminate the need for Non-Maximum Suppression (NMS). While subsequent iterations like Deformable DETR

[48] improved convergence through sparse sampling and RT-DETR [49] optimized encoders for real-time speed, RF-DETR is structurally derived from LW-DETR [16]. It adopts a modular encoder-projector-decoder architecture that is further optimized via Neural Architecture Search (NAS) to identify Pareto-optimal configurations by varying patch-sizes, the number of decoder layers, the number of queries, image resolution, and the number of windows in the power attention block [47].

The architecture (see Figure 2.4 for a full view) begins with the encoder, which diverges from standard ViT backbones by utilizing DINOv2 [18]. This self-supervised Vision Transformer is pre-trained on massive curated datasets to yield robust, "all-purpose" visual features and is processed using efficient windowed self-attention to manage computational costs[50]. Linking the encoder and decoder, and serving as the neck, is the projector. The projector employs C2f blocks adapted from YOLOv8 to fuse multi-scale features effectively. The decoder implements a mixed query selection strategy [51] to improve initialisation. In standard DETR models, object queries are typically static, learnable embeddings that must learn to locate objects from scratch. In contrast, this strategy extracts the top-K features from the projector's last layer representing the regions with the highest probability of containing objects and uses their positions to dynamically initialize the spatial queries. These positional priors are then combined with learnable content queries. Essentially, this gives the decoder a 'head start' by explicitly pointing it toward relevant image regions, which significantly accelerates training convergence (slightly functioning as a prior).

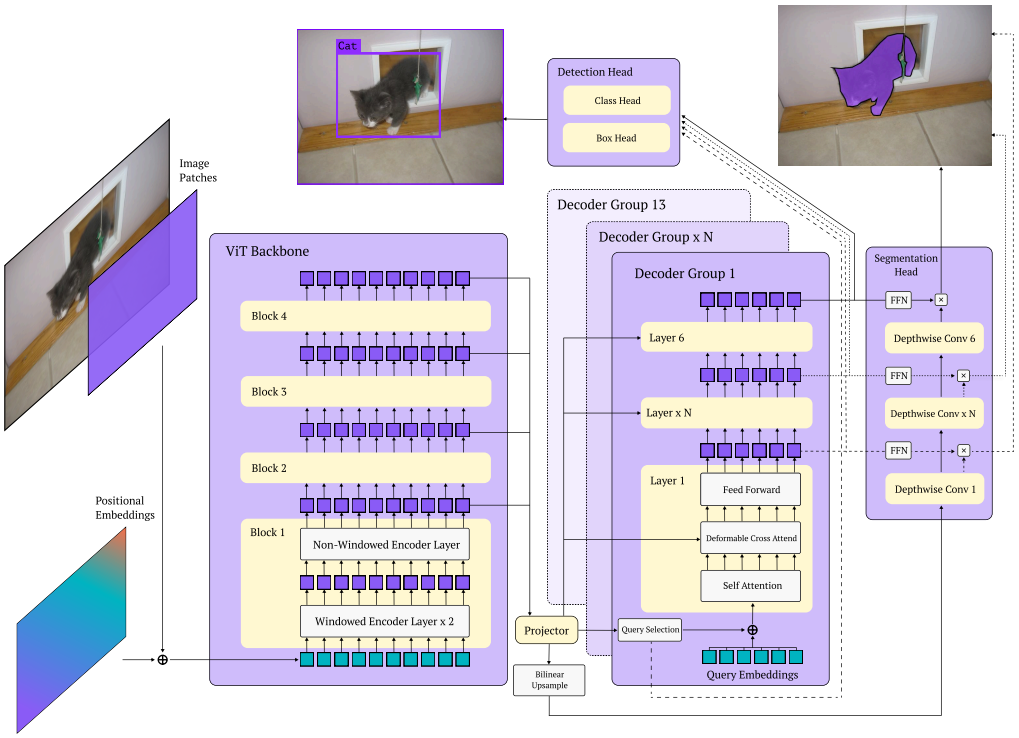


Figure 2.4: Overview of the RF-DETR architecture utilizing DINOv2 and NAS [47].

To stabilize and accelerate convergence, RF-DETR adopts Group-DETR [52] training

dynamics. The object queries are instantiated as K parallel groups, where each group undergoes independent one-to-one bipartite matching (the linear sum assignment problem), which is solved using a variation of the Hungarian algorithm. This effectively creates a global one-to-many assignment strategy where a single ground-truth object serves as a positive target for K predictions (one per group), significantly increasing the density of supervision signals per image. This auxiliary grouping is utilized solely for training, during inference, the extra groups are discarded to revert to a single-decoder architecture. Although there are K decoder groups during training, they all share weights, thus minimally increasing memory [52]. The model minimizes a composite loss L_{total} applied to all intermediate decoder layers and encoder proposals. The classification term, L_{cls}, utilizes an IoU-aware Binary Cross-Entropy loss. Instead of a static binary label, the target for positive samples is softened to a dynamic quality score $t = p^{\alpha} \cdot \text{IoU}^{1-\alpha}$, where p is the predicted probability of that class, IoU is the intersection-over-union with the ground truth, and α is a hyperparameter that acts as a balancing coefficient for localisation and class confidence. This formulation aligns classification confidence with localisation accuracy, explicitly suppressing high-confidence but poorly localized predictions. Simultaneously, the box regression terms combine a standard $L1$ loss for absolute coordinate accuracy with a Generalized IoU (GIoU) loss [53], which ensures non-vanishing gradients even for non-overlapping boxes by penalizing the smallest enclosing rectangle.

2.5 Self-supervised learning and domain adaptation in object detection

Modern detectors increasingly rely on self-supervised (label-free) pretraining to improve downstream transfer under distribution shift. Contrastive pretraining (e.g., MoCo [54]) learns instance-discriminative representations without labels, while vision transformers trained with self-distillation (DINOv2) [18] or masked image modeling (MAE) further boost robustness to domain changes when fine-tuned for detection [55]. These paradigms reduce reliance on labeled source domains and provide stronger initialisation for adaptation.

Automatic adaptation for certain domains has improved significantly compared to standard source-only baselines. RoyChowdhury et al. [56] demonstrated that automatically obtaining pseudo-labels from the source (or "base") detector and refining them with single-direction tracking allows the model to self-train effectively. This process fills in missing detections (false negatives) in the new domain, leading to a promising performance increase over the original pre-trained model.

In the self-driving vehicle domain, a novel semi-supervised training method was integrated into YOLOv5 that improves label generation by utilizing both high- and low-confidence predictions, rather than discarding the latter. The authors introduce a bi-directional object tracking mechanism that leverages temporal data (past and

future frames) to refine bounding boxes and recover missing labels [57]. However, the core idea of using low-confidence labels is the innovation behind the "Byte-tracker", which was published in 2022, a simple yet robust multi-object tracking method. By associating high-score boxes first and then leveraging similarities between low-score boxes (often occluded objects) and existing tracklets, the method recovers true objects and filters background noise [58].

Closest to surveillance deployments, self-supervised scene adaptation specializes a generic detector for a single fixed-view camera using only its unlabeled stream. Zhang & Hoai [21] propose cross-teaching between two base detectors, bidirectional tracking for pseudo-label densification, location-aware mixup that respects fixed object priors, and explicit background modeling/fusion. They introduce Scenes100, a 100-camera benchmark and evaluation protocol for per-scene adaptation. This line shows sizable accuracy gains without any human annotation in the target scene [21]. Finally, label fusion is managed by a graph-based refinement module that acts as a 'consensus engine' to eliminate duplicates. By treating every candidate box as a graph node and drawing edges between overlapping predictions, the module constructs connected components representing single objects. From each component, it selects the node with the highest degree—the box that overlaps with the greatest number of other predictions—thereby consolidating the most consistent spatial proposal from the combined detector and tracker streams, functioning as a "consensus engine".

Independent of adaptation, environmental/context specialisation can lower intra-scene variability. By routing images to indoor versus outdoor expert models, they observe statistically significant mean average Precision (mAP) gains over a single generalist network—showing that straightforward scene categorisation (e.g., a Places-based router) provides benefits that are complementary to self-supervised adaptation [59].

To address the challenges of detecting small objects in high-resolution imagery—such as drone or satellite surveillance where targets often lack pixel detail, Akyon et al. proposed Slicing Aided Hyper Inference (SAHI) [60]. This method employs a slicing strategy during both fine-tuning (by augmenting data with zoomed-in patches) and inference (by processing the image in overlapping slices). By recovering small details before merging the results, SAHI proves highly effective for scenarios involving small, densely packed targets.

2.6 Evaluation metrics

The industry standard evaluation metrics that are most commonly used is mAP this metric incorporates both localisation and classification to determine a true positive. It is calculated using IoU. which quantifies the spatial accuracy of a predicted bounding box relative to the ground truth. It is calculated as the ratio of the overlapping area

between the predicted box (B_p) and the ground truth box (B_{gt}) to the total area covered by their union.

$$IoU = \frac{\text{Area}(B_p \cap B_{gt})}{\text{Area}(B_p \cup B_{gt}}$$

A prediction is classified as a True Positive (TP) only if its IoU with a ground truth object exceeds a specific threshold (e.g., IoU $\geqq$ 0.5) and the class labels match. Predictions falling below this threshold, or duplicate detections of the same object, are penalized as False Positives (FP).

I compute the mean average precision, which aggregates the area under the Precision-Recall (PR) curve across all classes. Precision measures the purity of positive predictions, while Recall measures the proportion of ground truth objects that are successfully detected.

Crucially, the PR curve is generated by ranking all detections by their confidence score (from highest to lowest). The final AP is derived using maximum interpolation, where the precision at a given recall level r is taken as the maximum precision for any recall $r' \geq r$. This interpolation ensures that the metric rewards the model for placing correct detections at the top of the ranking and mitigates penalties for low-confidence false positives once all ground truth objects have been recalled mAP_{50}: This is computed as the mean Average Precision at a single IoU threshold of 0.50. $mAP_{50:95}$: This metric, which is the main standard used in industry, is obtained by averaging the average precision over 10 IoU thresholds, ranging from 0.50 to 0.95 in increments of 0.05.

2.7 Establisehd comparisons and peformances

A study by Sapkota et al. [17] directly evaluated RF-DETR against YOLO12 for greenfruit detection in complex orchard environments. The results highlighted RF-DETR's superior localisation in single-class settings (mAP@50 = 0.9464) and robust performance in multi-class occluded scenarios (mAP@50 = 0.8298). Qualitative analysis further demonstrated that RF-DETR's global self-attention mechanism allowed it to recover heavily occluded or camouflaged objects more effectively than YOLO12, which tended to over-detect in cluttered regions. Furthermore, RF-DETR exhibited significantly faster convergence, plateauing in fewer than 10–20 epochs, validating the advantage of its pre-trained DINOv2 backbone.

Recent benchmarks reveal a critical divergence between parameter efficiency (model size) and latency efficiency (inference speed). While the YOLO11 family retains a distinct advantage in pure storage requirements—YOLO11-N (2.6M params) is

nearly 12× smaller than RF-DETR-N (30.5M params)—this size advantage does not translate to superior runtime performance. Instead, transformer-based models (LW-DETR and RF-DETR) establish a great performance on Accuracy-Latency curves, although a new method is challenging the field.

For applications constrained by inference time rather than memory efficiency, RF-DETR offers significantly higher accuracy per millisecond of compute. Notably, RF-DETR-N matches the latency of YOLO11-N (≃2.3 ms vs 2.2 ms) but delivers a massive +10.9 mAP improvement on COCO (48.0 vs. 37.1). This indicates that while transformers require more memory to store weights, their parallelizable architecture allows them to process information as fast as much smaller, deeper CNNs while extracting far richer features. In the high-accuracy regime (>5 ms), the RF-DETR family remains optimal, with the RF-DETR-2XL achieving the highest accuracy across all tested benchmarks [47].

Table 2.1: Comparison of YOLO11, LW-DETR and RF-DETR variants on COCO and RF100-VL.

Family	Variant	Params (M)	Latency (ms)	AP_{COCO}	AP^{50}_{COCO}	AP_{RF100}	AP^{50}_{RF100}
YOLO11	N	**2.6**	2.2	37.1	51.6	55.5	81.3
	S	9.4	3.2	44.1	59.3	56.4	82.5
	M	20.1	5.1	48.3	63.6	57.0	82.5
	L*	25.3	6.2	53.4	–	–	–
LW-DETR	N	12.1	**1.9**	42.9	60.7	57.1	84.7
	S	14.6	2.6	48.0	66.8	57.4	85.0
	M	28.2	4.4	52.6	72.0	59.8	86.8
RF-DETR	N	30.5	2.3	48.0	67.0	57.6	84.9
	S	32.1	3.5	52.9	71.9	60.7	87.0
	M	33.7	4.4	54.7	73.5	61.5	87.7
	L†	135.6	–	59.0	77.3	–	–
	2XL	126.9	17.2	**60.1**	**78.5**	**63.3**	**88.9**

General: COCO YOLO11-N/S/M and all LW/RF-DETR (N/S/M/2XL) metrics are retrieved from the RF-DETR paper [47].

* Metrics from official Ultralytics documentation; RF100-VL metrics were not reported.

† RF-DETR-L (preview) parameters and COCO AP / AP^{50} are reported by Roboflow (model zoo and GitHub); latency and RF100-VL metrics were not reported.

3

Methods

This chapter describes the methodology required to address the primary aim of evaluating self-supervised scene adaptation in edge-computing environments.

To investigate the trade-offs between YOLO11 (CNN) and RF-DETR (Transformer) models in a fixed camera scene, the dataset used is of utmost importance. Generic object detection benchmarks do not adequately capture the stationary backgrounds, specific camera angles, and environmental noise inherent to surveillance. Therefore, I will first introduce the Scenes100 dataset, chosen specifically to emulate diverse, realistic fixed-camera environments. This data foundation is essential for correctly evaluating how well different model sizes and architectures can adapt to real-world deployment.

Secondly, an objective is to determine if on-device adaptation can compete with resource-heavy methods. Consequently, I outline four distinct label generation strategies. These strategies are selected to establish the potential and limitations of the proposed method: a naive baseline, a heavy real-time ensemble, the proposed resource-efficient SAHI+ByteTrack, and a general auto-labeling server-side model using SegmentAnythingModel3 (SAM3).

Thirdly, to determine if the specific static characteristic of surveillance video can be leveraged to improve performance, I describe the implementation of two different model adaptation approaches: Standard fine-tuning and a modified background-context fusion method derived from the method by Zhang et al [21]. This section also details the strict freezing of backbones to safeguard for catastrophic forgetting.

To ensure that the findings are robust against real-world environmental shifts, I introduce a method for seasonal data creation using generative AI to understand

how performance may vary as the environment shifts and what strategy should be used in a production setting.

Finally, I describe the validation metrics (mAP, IoU) used to quantify performance. In Figure 3.1 a general overview of the experiments is shown.

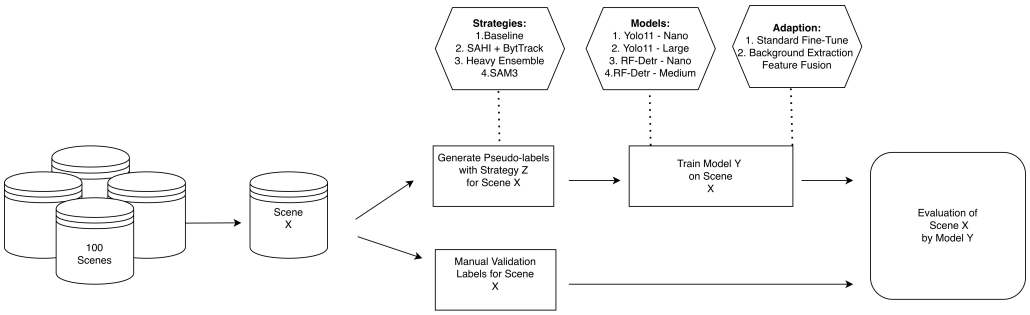


Figure 3.1: Overview of the general approach of the Experiments

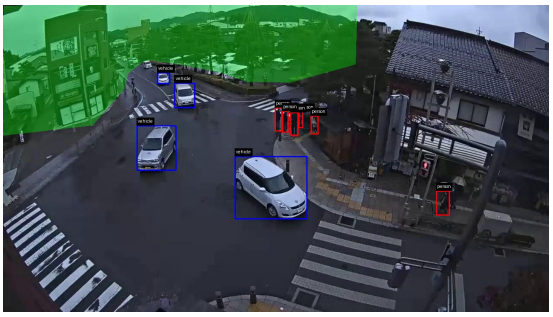
3.1 Dataset and models

This section describes the dataset and object detection models used to evaluate self-supervised adaptation. A geographically diverse fixed-camera dataset is combined with models of varying capacity to study how architecture and computational budget affect adaptation performance, particularly for edge deployment.

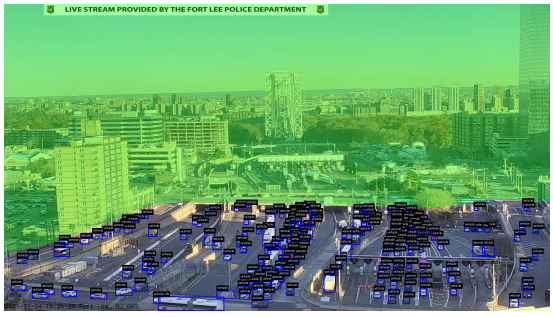
3.1.1 Dataset Scenes100

To evaluate the model's performance across diverse surveillance environments, I utilize the Scenes100 dataset [21]. This dataset serves as a benchmark for scene-adaptive object detection, consisting of 100 distinct videos captured from fixed-perspective cameras across 16 countries. The videos capture a wide variety of environments, ranging from crowded urban centers to isolated roadways, covering different times of day, weather conditions, and object densities. The dataset targets two primary categories: person and vehicle. The vehicle category includes all vehicles with four or more wheels and thus corresponds to the COCO categories: car, bus, and truck. Crucially, each scene includes manually annotated evaluation frames and a spatial validity mask. This mask is applied to filter out irrelevant regions, such as distant backgrounds. The number of validation frames per scene varies, scenes with a dense number of objects will have fewer validation frames. For my experiments, I adopt the training frame splits provided by the official implementation [21]. However, to reduce computational overhead, I limit the training data to the last 9,000 samples of the provided sequence. These frames are extracted at a stride of 5 (every 5th frame) from the original 30 FPS videos, the 9000 frames represents 30 minutes of video. The resolution varies from 720 $\times$1280 to 1080$\times$1920. Four examples of the scenes are shown in Figure 3.2.

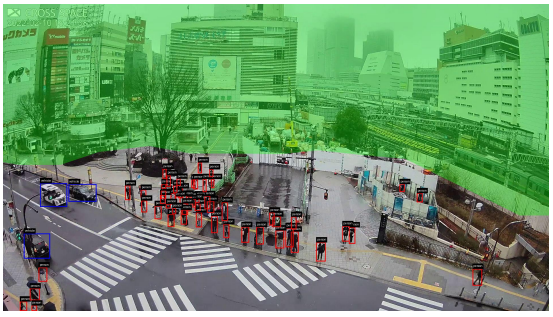
(a) Scene 001



(b) Scene 003



(c) Scene 019)



(d) Scene 146

Figure 3.2: Examples of diverse surveillance scenes from the Scenes100 dataset, with ground truth labels, red boxes are "person" and blue boxes are "vehicle". The see-through green overlay is the validation mask (no objects in this area will be included).

3.1.2 Models

To address evaluation of self-supervised adaptation across different architectural paradigms and model capacities, four distinct models were selected. The selection criteria focused on representing the current state-of-the-art for both convolutional Neural Networks (CNNs) and Real-Time Detection Transformers (RF-DETRs).

I selected four models representing the current state-of-the-art. The **YOLO11** (Nano and Large) serves as the CNN representative, the Nano variant tests adaptation under extreme edge constraints, while the Large variant establishes a performance ceiling. These are contrasted against the **RF-DETR** (Nano and Medium), representing the Transformer architecture. Notably, RF-DETR-Medium is selected over the Large variant to maintain strict hardware constraints in the edge-environment. The Medium version of the RF-DETR is also closer in both parameters and latency to Yolo11-Large than what the RF-DETR-Large version. Thus, it will allow for a fairer comparison. A full comparison of models can be seen in Table 2.1.

3.2 Pseudo-Label Generation Strategies

To investigate whether on-device training is competitive with heavier methods, this section outlines four pseudo-label generation strategies. These strategies are:

1. SSL-B (Self-Supervised Learning Baseline)
2. Ensemble (Real-time ensemble)
3. ST (SAHI + ByteTrack)
4. SAM3 (Server-based SegmentAnythingModel3)

All models are pretrained on 640×640 resolution, which is used unless otherwise specified. The resulting pseudo-labels act as ground truth for training in the respective strategy.

3.2.1 Baseline 1: Self-Supervised Learning Baseline (SSL-B)

This strategy represents the naive baseline where a COCO-pretrained detector generates pseudo-labels without any refinement. A strict confidence threshold of $\lambda_{\text{det}} = 0.5$ is applied to filter initial predictions. Additionally, for YOLO-base detectors, Non-Maximum Suppression (NMS) is utilized with an IoU threshold of $\lambda_{\text{nms}} = 0.75$ (the Ultralytics default) to eliminate redundant detections.

3.2.2 Heavy Real-Time Ensemble (Ensemble)

Following the self-supervised scene adaptation framework proposed by Zhang et al. [21], this strategy generates high-quality pseudo-labels utilizing a computationally

intensive ensemble strategy. It represents the upper limit for what we can count as real-time edge deployment due to very high hardware needs. This "heavy" approach serves as a robust baseline. The pipeline aggregates predictions from two large-scale detection models, RF-DETR-Large and YOLO11-X (X-Large).

These initial detections are subsequently refined via bi-directional tracking using DiMP50 (Discriminative Model Prediction) [61]. DiMP50 is a powerful single-object tracker that learns a discriminative target model to distinguish objects from the background. In the pipeline of Zhang et al. [21], they initialize the tracker using deep features extracted from the detection boxes (utilizing the ResNet-50 backbone) and propagate these candidates in both forward and backward temporal directions. This bi-directional strategy is crucial for recovering false negatives in adjacent frames and refining localisation accuracy via the tracker's precise IoU estimation component.

Final candidates are determined through a graph-based merging step, where boxes with identical class labels and high intersection-over-union (λ_{iou}) are combined, retaining only the most connected candidate. I adopt the well-performing hyperparameters established by Zhang et al. [21].

3.2.3 Proposed Method: SAHI + ByteTrack (ST)

I propose a resource-efficient pipeline designed to take advantage of the temporal data of videos and the pretrained model. The method leverages Slicing Aided Hyper Inference (SAHI) [60] to improve small-object performance and ByteTrack [58] to recover low-confidence detections temporally. The main idea behind this is that if one can run a model in their edge-environment for inference, the creation of the pseudo-labels should also be possible in the same edge-environment.

1. **Global and Local Inference:** Standard inference is first run on the full frame to capture global context. In parallel (if applicable), SAHI performs inference on overlapping windows (overlap ratio 0.1). Window sizes are set to 640×640.

2. **Hierarchical Merging:** To prevent object fragmentation (where a single large object is detected as multiple parts across windows), **global detections are prioritized**, to make tracking and labels more stable. If a SAHI detection is contained within a high-confidence global detection, with more than a certain % of the total area, the global box is retained, and the SAHI-based box is discarded. Then I perform NMS to get remove potential duplicates.

3. **Temporal Recovery (ByteTrack):** Firstly, I save all high confidence labels at this stage, then I utilize a bi-directional implementation of the ByteTrack algorithm [58] to mitigate trajectory fragmentation caused by occlusion or

motion blur. Two copies are created, one copy stays in current temporal order while the other copy is reversed, thus we have a labels for both temporal directions. Then I use the ByteTrack algorithm to recover labels. Unlike traditional tracking methods that strictly discard detections below a high confidence threshold (e.g., $\lambda_{\text{high}} > 0.5$), ByteTrack employs a hierarchical data association strategy.

- **First Association:** High-confidence detections are initially matched to existing tracklets using Kalman Filter motion predictions and Intersection-over-Union (IoU). If there is a high confidence label in the frame not being matched from a previous frames, we initiate a tracklet.

- **Second Association:** Crucially, any tracklets that remain unmatched are not immediately terminated. Instead, the algorithm searches a secondary pool of low-confidence proposals (threshold $0.01 < \lambda_{\text{low}} < 0.5$) to find spatial matches.

Thus, based on the active tracklets, I can recover some potential low-confidence labels. However, as I perform bi-directional tracking, some labels can be recovered twice, thus I run NMS across the recovered labels to remove duplicates. The whole flow of creation of pseudo-labels is visualised in Figure 3.3.

3.2.4 Server-based (SAM3)

This strategy yields a high-quality labeled dataset and serves as a representative commercial-grade labeling approach. It leverages the **SAM 3 Video** tracker [62] as a general-purpose, server-side commercial auto-labeling tool. What makes SAM3 a great choice for creating labels for videos is that the model has a detector and tracker implemented, thus, one can create high-end labels end-to-end. In my implementation, SAM3 is prompted with text concepts to map outputs to our specific ontology: person uses the prompt person, while vehicle aggregates prompts for car, truck, bus. To stabilize tracking while avoiding duplicates, frames are processed in overlapping sliding windows containing 35 saved frames plus 2 looking-back to intialise tracking context frames. Finally, per-frame outputs are filtered to retain boxes with scores $> \lambda_{\text{Sam3}} = 0.5$ and aggregated via NMS ($\lambda_{\text{nms}} = 0.85$) to remove duplicates.For this strategy, I use the original resolution of the videos for full performance.

3.3 Training details and adaptation strategies

This section describes the implementation of the experiments. Firstly, I will cover the two adaptation strategies for the base-models: Standard fine-tuning and background-context fusion. Finally, I will discuss how the general training is conducted.

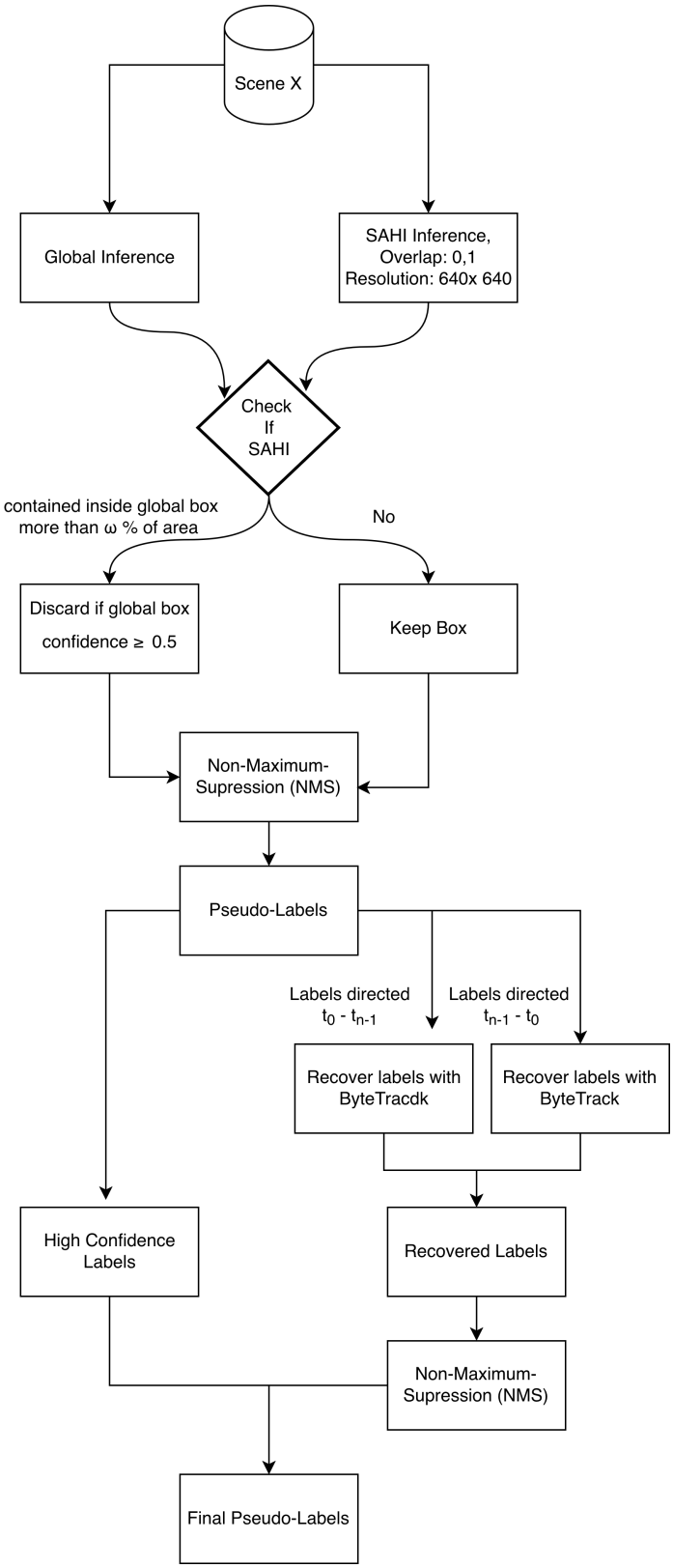


Figure 3.3: Overview of the flow for creation of pseudo-labels using SAHI + ByteTrack (SBT).

3.3.1 Standard Fine-Tuning (SF)

This serves as the baseline adaptation method. The method is a standard fine-tuning strategy; I only re-initialize the head and adapt the model for the input resolution:

- **Initialisation:** Models are initialized with COCO pre-trained weights. For RF-DETR, positional encodings are bilinearly interpolated to 640×640.

- **Head re-initialisation:** The classification and regression heads are re-initialized to facilitate the fewer classes.

- **Training:** The backbone remains frozen. A constant learning rate and fixed batch size are applied across all experiments to ensure comparable convergence.

3.3.2 Background-Context Fusion (BF)

To explicitly leverage the static nature of surveillance cameras, I adapt the background-fusion concept from Zhang et al.[21] which is most similar to their "mid-fusion" adaptation. However, with two main differences, the backbone is frozen and I do not add a loss-function to regularize the update of backbone weights. The flow of the modified architecture can be seen in Figure 3.4.

- **Background generation:** A dynamic background reference image B is constructed by temporal aggregation. I used the already created background images from the official repository of Scenes100. [21], thus this implies I borrow the theoretical method of creating them as well, which is as follows. For a video frame I of dimension $H \times W$ associated with a set of K pseudo-annotated object bounding boxes $\{(x_{1,k}, y_{1,k}, x_{2,k}, y_{2,k}) \mid k = 1, \ldots, K\}$, a background mask M of the same dimension as I can be constructed as follows. For each pixel (x, y) in M, set $M[x, y] = 0$ if (x, y) is inside of any pseudo-annotated bounding box, and 1 otherwise. Then, for a sequence of frame-mask pairs $\{(I_l, M_l) \mid l = 1, \ldots, L\}$, the background image is determined as:

$$B = \frac{\sum_{l=1}^{L} I_l \otimes M_l}{\sum_{l=1}^{L} M_l}, \tag{3.1}$$

  where $\otimes$ is the pixel-wise multiplication operator.

  However, there might be a location (x', y') that lies inside an object bounding

box in every image, for example, a parked car, $M_l[x', y'] = 0$ for all l. In this case, the background at this location is never observed, and its pixel value cannot be determined via Equation 3.1. In those cases, an inpainting algorithm was used [21].

- **Object mask:** An object mask M_O is derived via difference after normalisation ($M_O = (I - B + 1) \times 0.5$) to serve as the secondary input stream. The mapping of the background to the current frame in training is set to "nearest" in the temporal aspect and validation, thus providing the closest match for the fusion.

- **Parallel streams:** The original image I and the Object Mask M_O are processed through two parallel, frozen backbones.

- **Feature-level fusion:** Unlike prior works that use dual-branch losses to update the backbone, I strictly keep the backbone frozen. Fusion occurs at the scalar level, thus, feature maps from the image stream and mask stream are combined via **element-wise averaging**.

- **Prediction:** The fused multi-scale feature map is passed to the detector neck and head.

3.3.3 Training setup

For each generated set of pseudo-labels, the models will be trained using the official repositories, which I have adapted to fit the current settings. All models have been pretrained using the COCO dataset, and been provided by the models open-source repositories [63], [64]. These are the models that I will refer to as the base models.

To ensure similar settings for training, I use the AdamW optimiser with a constant learning-rate of $1 \cdot 10^{-4}$. A duration of 2 epochs is used to train the models. No augmentations, except for resizing and normalizing, are taking place. Resolution for training and inference is 640×640. A batch size of 12 is used, and the final model being used for validation after 2 epochs is the Exponential Moving Average of Weights (EMA) model, using the default settings of their respective public repositories.

Once training begins, consistent with modern literature on avoiding catastrophic forgetting in foundation models [21], [65], all backbones are strictly frozen during the training process. All other parameters are trained during the experiments. See Table 3.1 for specifics of parameters and trainable parameters. For RF-DETR, the pretrained models have not been trained using the resolution 640×640 as Yolo11 has been; for medium, 576×576 was used, and for nano, 384×384 was used. Thus,

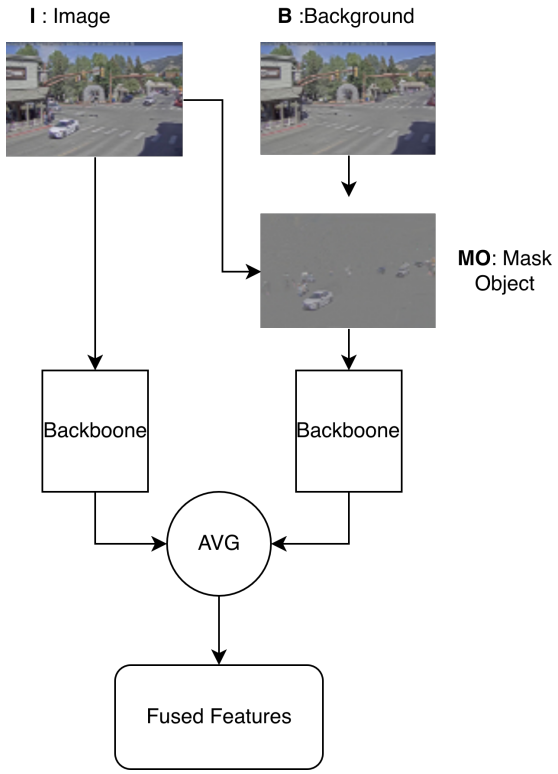


Figure 3.4: Overview of the modification in Background-Context Fusion from flow of input until the output of the fused features to further be proccesses

the positional encodings will be bi-linearly interpolated to fit 640×640 resolution.

Furthermore, YOLO has been trained using Letter-boxing [45], retaining the aspect ratio in the image resizing, while RF-DETR does not. I have kept this behavior consistent with the pre-training procedure, since altering it led to a substantial drop in performance.

Table 3.1: Model Parameter Breakdown (in Millions)

Model	**Total (M)**	**Trainable (M)**	**Frozen (M)**	**% Trainable**
YOLO11n	2.590	1.225	1.365	47.3%
YOLO11l	25.312	12.478	12.834	49.3%
RF-nano	30.467	6.885	23.583	22.6%
RF-medium	33.687	9.828	23.859	29.2%

3.4 Seasonal Data creation

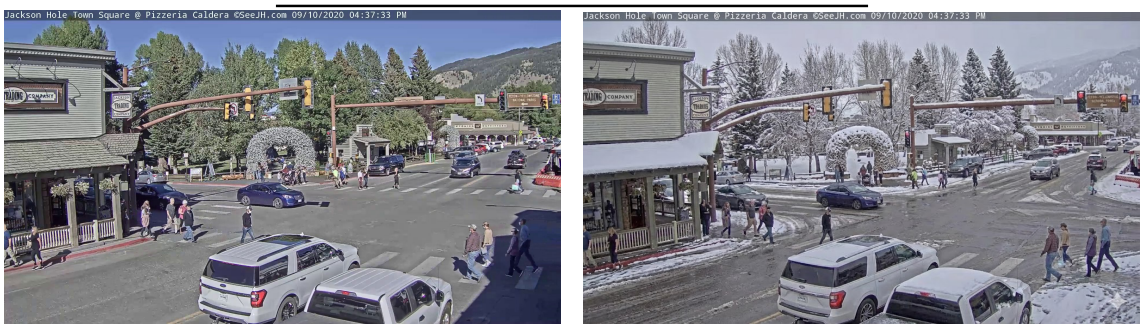
To understand how a model trained for a specific scene responds to environmental changes when the scene's conditions shift during real-world deployment. I chose to translate one scene filmed during the summer into a winter session. This is done by using Nano-banana-Pro from Google [66] which is a multi-modal-to-image model. By

uploading one frame at the time of the validation frames (the summer/original image) and a prompting. The prompt was subjected different minor changes.However, these adjustments were not explored over an extended period of time. It was selected based on my perceived similarity among the generated images during visual inspection.
"I am using a validation set in the summer for a static camera surveillance object detector. However, I would like to create a winter version of it, Object location cant be changed no matter what. Do it for this, make sure not to ADD/REMOVE or CHANGE people or cars locations" [66] .

The selected scene for the experiment is Scene 001, and a side-by-side comparison example from this scene is shown in Figure 3.5. However, not all objects remained consistent after generating the winter version of the frame, and as a result, it was necessary to manually re-annotate certain frames. The total number of objects decreased for the person class and increased for the vehicle classes across all validation frames when comparing the original summer images to the generated winter versions. The total changes in the ground-truth objects are presented in Table 3.2, representing the total number of objects across all validation frames from the person class and the vehicle class.

Table 3.2: Change in Ground Truth Labels by Season

Class	**Summer (original)**	**Winter**
Person	485	344
Vehicle	221	229



(a) Original image (b) Generated image from Nano-banana-pro

Figure 3.5: A side-by-side comparison. On the left (a), we see the original summer image, while the right (b) shows the generated winter image

3.5 Validation

This section outlines how the evaluation will be carried out and which metrics will be reported. As introduced in the preliminaries, the primary metric is the industry-standard mean Average Precision (mAP). In line with the COCO evaluation protocol, I restrict evaluation to the top 100 detections per image, sorted by confidence score. This ranking procedure ensures that evaluation focuses on the model's most confident

predictions. Because average precision uses maximum interpolation, low-confidence detections that exceed the number of ground-truth instances (i.e., the "tail") do not reduce the final score, as long as true positives appear at the top of the ranking. This yields a standardized evaluation setting that avoids unfairly penalizing the model for low-confidence background noise in sparse scenes, and follows the standard COCO evaluation protocol.

Following the COCO evaluation protocol, I also report AP broken down by object size: AP_{small}, $\text{AP}_{\text{medium}}$, and AP_{large}. These metrics are computed based on the ground-truth bounding box area ($w \times h$ in pixels), thus the original resolution of the videos:

- AP_{small}: Objects with area $< 32^2$ pixels (i.e., smaller than approximately 32×32 pixels).
- $\text{AP}_{\text{medium}}$: Objects with area in the range $[32^2, 96^2)$ pixels.
- AP_{large}: Objects with area $\geq 96^2$ pixels (i.e., larger than approximately 96×96 pixels).

For the base-detectors to align with the surveillance context, the COCO class ontology is remapped to person and vehicle, with classes mentioned in Section 3.1. Additionally, since the number of persons and vehicles varies across scenes in the validation sets, I compute a weighted mAP based on the ratio of the total number of persons to vehicles in each scene. This metric is reported to provide a more fair and representative evaluation of scene-level performance.

Each fine-tuned model is evaluated on the specific scene for which it was adapted. This results in a total of $100 \times 4 \times 2 \times 4 = 3200$ models to be evaluated, covering all combinations of scenes, model architectures, adaptation methods, and pseudo-labeling strategies. Furthermore, following the practice of Zhang et al. [21], I apply the non-evaluation mask; the bounding boxes that have at least one corner inside the non-evaluation mask will be removed. Thus, distant parts in the frames where objects are deemed too small and blurry will not affect the evaluation results. The average metrics across scenes will be reported.

When referring to AP solely in the coming chapters, I will be referring to the $AP_{50:95}$.

4

Results

This section goes through the results from the experiments, starting with the performance across the 100scenes dataset and how the different architectures converged. Further, I will look deeper into how the different pseudo-labeling strategies perform. Additionally, I will cover results of the different Adaptation strategies, and Background-Context Fusion compared to Standard-Finetuning. Lastly, I will report the results of the seasonality changes impact on the models performances.

4.1 Architecture and model-size

This section will address the impact of architecture choice and model size on detection performance and their adaptation capabilities models are compared across nano and medium/large scales, using both base-models (COCO-pretrained) and adapted configurations. Performance is evaluated quantitatively using weighted and raw AP metrics, complemented by qualitative cases and an analysis of convergence behavior.

First section of Table 4.1 reports the performance of the base models (COCO-pretrained) across all scenes in the scenes100 dataset. The RF-DETR Medium model achieved the highest overall performance of 0.4535 AP_{weighted}, whereas the YOLO11-nano model attained the lowest at 0.2406 AP_{weighted}.

The second section of Table 4.1 provides an overview of the most promising model configurations of both architectures and sizes, chosen based on the highest performance of AP_{weighted}. SAM3 is included in all configurations except one; the configuration that does not incorporate SAM3 instead employs the ST strategy. The best-preforming configurations for the RF-DETR models both utilize the background-context fusion strategy, whereas both YOLO-based models rely on standard finetuning. Further, the RF-DETR models outperform the YOLO11 variants across metrics. The best RF-DETR achieving 0.4912 AP_{weighted} while the best YOLO11 achieves 0.4721 AP_{weighted}.

Compared to the base performance, we can see that RF-DETR Medium still achieves the highest performance and that RF-DETR Nano has surpassed YOLO11-large. Notably, the performance improvement of YOLO11-Nano is substantially greater than that of the other models, going from 0.2406 $\Rightarrow$ 0.375 AP_{weighted} exhibiting an increase of more than 50% relative to its baseline configuration.

Based on qualitative analysis, certain scenes exhibit very low performance across all models, scenes which can be characterized by a dense amount of small objects where the camera is fixed far from the actual objects. One example of this is scene 019, which can be seen in Figure 4.1, where the best performing model configuration is RF-DETR medium using BF and SAM3 reached 0.22 AP_{weighted} while the worst model configuration was YOLO11-Nano which can not detect any objects.

Table 4.1: Performance comparison between base models and best adapted configurations (Mean over all scenes).

Model	Model Adaptation	Labeling Strategy	AP (Weighted)	AP50 (Weighted)	AP (Raw)	AP50 (Raw)
Base models (COCO pretrained, no adaptation)						
RF (Medium)	Base	None	**0.4535**	**0.7028**	**0.4431**	**0.6736**
RF (Nano)	Base	None	0.4113	0.6797	0.4037	0.6502
YOLO11 (Large)	Base	None	0.4227	0.6181	0.4059	0.5784
YOLO11 (Nano)	Base	None	0.2406	0.4107	0.2339	0.3832
Best adapted configurations						
RF (Medium)	Background-Context Fusion	ST	**0.4912**	0.7580	**0.4732**	0.7260
RF (Nano)	Background-Context Fusion	SAM3	0.4758	**0.7679**	0.4518	**0.7302**
YOLO11 (Large)	Standard Fine-tuning	SAM3	0.4721	0.7307	0.4393	0.6810
YOLO11 (Nano)	Standard Fine-tuning	SAM3	0.3750	0.6395	0.3416	0.5748

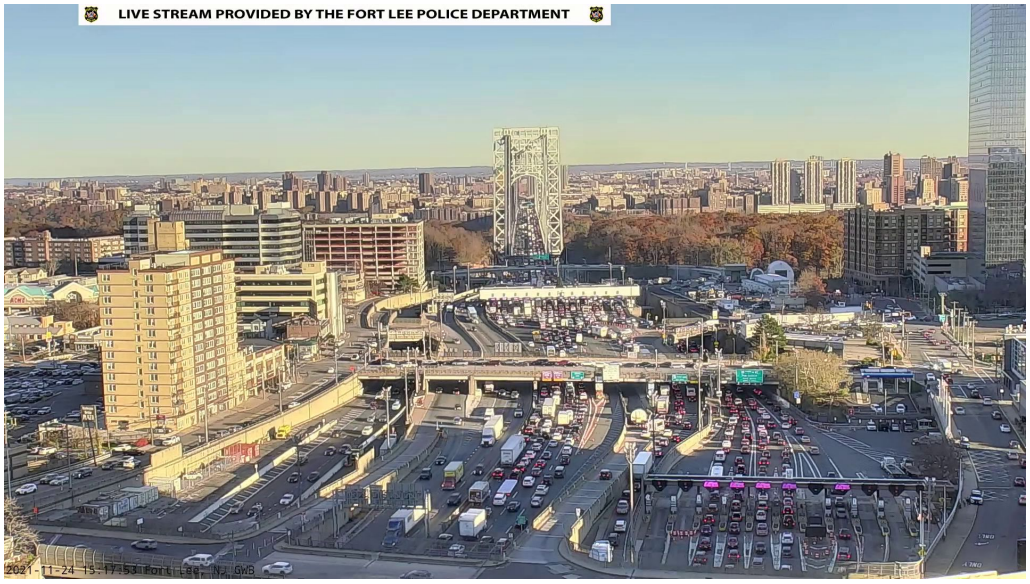


Figure 4.1: Image from Scene 019.

The convergence of the models respective rates of convergence during training, is illustrated in Figure 4.2. For each model–scene combination, the metric was first normalized and subsequently averaged over all scenes, providing a representative

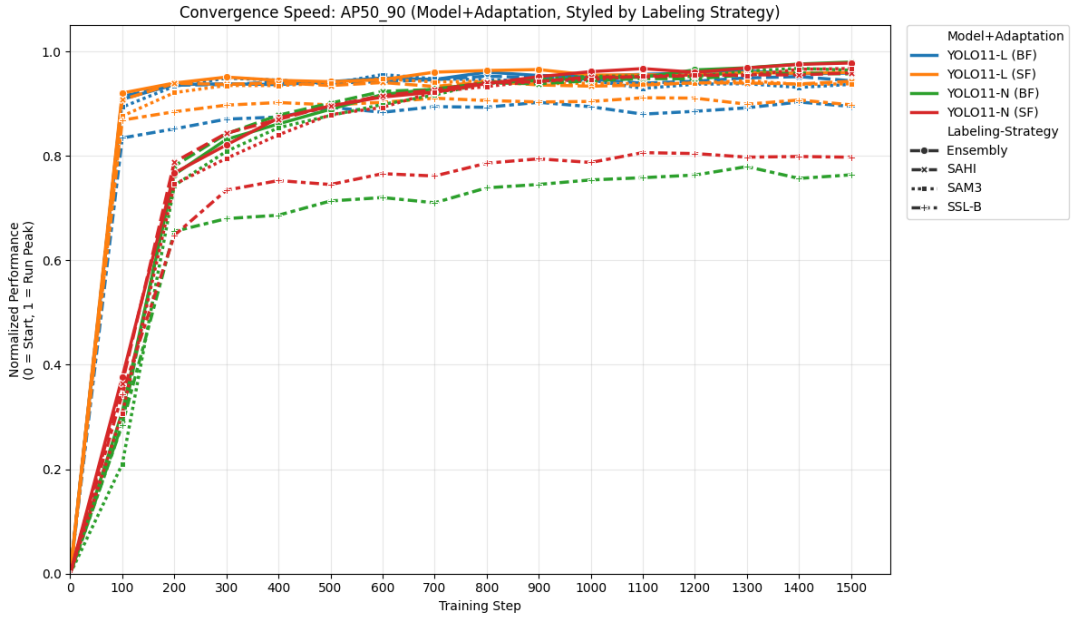
convergence curve for each model configuration. The RF-DETR model exhibits highly consistent convergence dynamics across different model variants, adaptation procedures, and labeling strategies. Its convergence is rapid, with performance beginning to plateau after approximately 200–300 batch steps. In contrast, the YOLO11-based models display greater variability. The YOLO11-Nano variants require more iterations to converge and the curve exhibit an approximately exponential convergence profile. The YOLO-Large model behaves more similarly to RF-DETR in terms of initial convergence speed. However, instead of reaching a clear plateau, it tends to continue to improve over a longer range of training steps, indicating more prolonged learning compared to RF-DETR.

4.2 Pseudo-labeling strategy

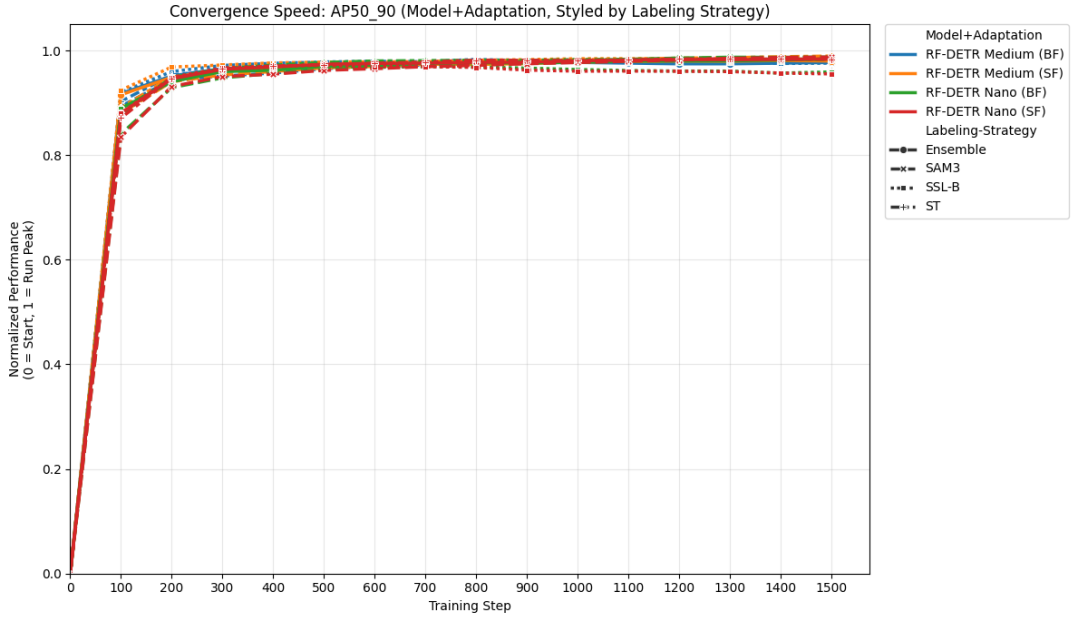
Table 4.2 presents the relative performance of each method with respect to its corresponding base model on the 100Scenes dataset. Several consistent patterns emerge:

First, the Self-Supervised Learning baseline (SSL-B), where the labels were created from the current model itself without any augmentation or improvements, does not improve performance for any model-configuration across all the scenes, while increases performance for some individual scenes (see APPENDIX). Furthermore, it leads to a pronounced degradation of the YOLO11-based models, YOLO11-Large decreasing $\sim 11\%$ in $AP_{weighted}$ compared to the RF-DETR-Medium decreasing $\sim 1\%$. Second, the proposed on-device strategy, the SAHI-ByteTrack (ST), yielded performance gains $AP_{weighted}$ across all methods and model families, being part of the best performing configuration shown in Table 4.1. The Smaller models in both architectures increased their performance relatively more from Ensemble and SAM3 than the larger ones. SAM3 also showed the largest and most consistent increase across model architectures and configurations. This is evident by the highlighted bold numbers, which represent the best labeling-strategy in all model adaptations seen in Table 4.2, except for the RF-DETR Medium Background-Context Fusion. Using of Ensemble strategy showed consistent increase for all metrics on YOLO11-Nano, while being more inconsistent for the other models.

Looking at the more detailed Table 4.3, a clear dichotomy emerges regarding class-specific improvements, particularly within the RF-DETR architecture. While the AP_{vehicle} scores remain relatively static across adaptation methods—hovering near the 0.51 baseline for the Medium model the AP_{person} metric demonstrates significant adaptation to scene adaptation, rising from a base of 0.3095 to over 0.37 in the best-performing configurations. Furthermore, a clear trend across both YOLO and RF-DETR architectures is that the most substantial relative gains are concentrated in the smaller object scales rather than the larger ones. While AP_{Large} shows only marginal improvements, often hitting a saturation point, AP_{Small} and AP_{Medium} exhibit dramatic increases; for instance, in the YOLO11-nano model, AP_{Small} nearly



(a) Convergence speed for YOLO11 models under Standard Fine-tuning (SF) and Background-Context Fusion (BF) and by labeling strategy.



(b) Convergence speed for RF-DETR models under Standard Fine-tuning (SF) and Background-Context Fusion (BF) and by labeling strategy.

Figure 4.2: Convergence dynamics across architectures. Curves show normalized AP50_90 performance over training steps, where 0 denotes run start and 1 denotes peak performance per run.

Table 4.2: Performance delta vs. base (percentage points). Positive values indicate improvement over the corresponding base model. Best values per model block are bolded.

Model	Model Adaptation	Labeling Strategy	Δ AP (Weighted)	Δ AP50 (Weighted)	Δ AP (Raw)	Δ AP50 (Raw)
RF (Medium)	Standard Fine-tuning	SSL-B	−1.33%	−3.31%	−1.84%	−3.50%
		ST	2.44%	3.11%	1.61%	2.45%
		Ensemble	0.08%	−3.79%	−0.44%	−3.33%
		SAM3	2.28%	5.43%	0.54%	3.61%
	Background-Context Fusion	SSL-B	−0.50%	−1.66%	−0.89%	−1.53%
		ST	**3.77**%	5.53%	**3.00**%	5.24%
		Ensemble	1.25%	−2.17%	0.72%	−1.53%
		SAM3	3.48%	**7.90**%	2.07%	**7.09**%
RF (Nano)	Standard Fine-tuning	SSL-B	−3.94%	−8.84%	−4.46%	−9.01%
		ST	2.58%	1.28%	1.44%	0.25%
		Ensemble	3.40%	−2.50%	2.55%	−2.07%
		SAM3	5.42%	6.58%	3.50%	5.12%
	Background-Context Fusion	SSL-B	−3.70%	−8.23%	−4.13%	−7.99%
		ST	3.61%	3.22%	2.40%	2.32%
		Ensemble	4.33%	−1.11%	3.60%	−0.30%
		SAM3	**6.46**%	**8.82**%	**4.81**%	**8.00**%
Yolo11 (Large)	Standard Fine-tuning	SSL-B	−11.69%	−22.28%	−10.69%	−19.32%
		ST	2.17%	0.64%	1.12%	0.65%
		Ensemble	−0.78%	−4.21%	−0.51%	−2.32%
		SAM3	**4.94**%	**11.26**%	**3.34**%	**10.26**%
	Background-Context Fusion	SSL-B	−11.70%	−22.44%	−10.62%	−19.21%
		ST	1.61%	−0.60%	0.74%	−0.42%
		Ensemble	−1.63%	−6.15%	−1.37%	−4.19%
		SAM3	4.45%	10.80%	2.88%	9.59%
Yolo11 (Nano)	Standard Fine-tuning	SSL-B	−7.43%	−16.51%	−7.36%	−15.04%
		ST	5.69%	7.20%	4.64%	6.12%
		Ensemble	10.41%	13.03%	9.13%	12.07%
		SAM3	**13.45**%	**22.88**%	**10.77**%	19.15%
	Background-Context Fusion	SSL-B	−6.78%	−15.75%	−6.94%	−14.63%
		ST	6.22%	7.54%	5.04%	6.44%
		Ensemble	10.17%	12.29%	8.74%	11.42%
		SAM3	13.42%	22.83%	10.71%	**19.20**%

triples under best configuration, going from $\sim 0.04 \Rightarrow 0.12$. We can also see in general that it is the smaller objects that are harder to detect, as the models consequently has worse score on the AP_{Small} metric.

4.3 Adaptation Strategy

Hinted at in previous sections, the Background-Context Fusion approach has a more pronounced positive effect on the RF-DETR models than on the YOLO11 models, being the best performing configuration of the RF-DETR models, while not being the best performing YOLO11 configurations.

An important goal is to determine whether background extraction and feature fusion lead to an improvement in performance,thus to further investigate and determine whether a statistically significant difference exists between Standard Finetuning (SF) and Background-context Fusion (BF). I employed a Wilcoxon signed-rank test. This non-parametric procedure was selected because visual inspection of the performance distributions indicated deviations from normality, including skewness, rendering the assumptions of the paired t-test questionable. The Wilcoxon signed-rank test is more robust under these conditions and therefore more appropriate for the analysis.

The corresponding results are reported in Table 4.4. For the RF-DETR models, the use of BF is clearly advantageous, regardless of the adaptation strategy, size, or pseudo-labeling strategy. YOLO11 shows a different outcome as it depends on the adaptation strategy, size, or pseudo-labeling strategy. YOLO11-Nano's significant results indicate that the BF appears to benefit from the less computationally intensive pseudo-labeling strategies (SSL-B, ST), whereas the non-significant outcomes remain either positive or ambivalent for the heavier models. In contrast, the significant findings for YOLO11-Large generally indicate a slight decrease in performance.

Nevertheless, specific scenes exhibit a systematic preference for particular methods. I highlight several such cases to further illustrate that the relative performance of the approaches can depend strongly on the nature and contextual characteristics of the scene. In Figure 4.3, scenes 156 and 058 are shown, where BF outperforms SF while also surpassing the base model. Conversely, Figure 4.4 presents scenes in which SF achieves superior performance compared to BF, again while exceeding the performance of the base model. For Scene 058, Yolo11-nano with BF reached 0.5123 $AP_{weighted}$ using SAM3 and 0.4676 using ST, while the YOLO11-Large base model reached 0.5202. This demonstrates that a model 10× smaller in parameters was able to achieve comparable performance.

Table 4.3: Detection performance across Scenes100 for each model, including base (non-adapted) performance, Standard Fine-tuning, and Background-Context Fusion. Values are mean across all scenes, AR being Average Recall at 100 Detections max. AP referring to $AP_{50:95}$

Model	Model Adaptation	Labeling Strategy	$\mathbf{AP_{person}}$	$\mathbf{AP_{vehicle}}$	$\mathbf{AR_{100}}$	$\mathbf{AP_{small}}$	$\mathbf{AP_{medium}}$	$\mathbf{AP_{large}}$
RF (Medium)	Base	None	0.3095	0.5150	0.5352	0.1810	0.5260	0.6651
	Standard Fine-tuning	SSL-B	0.2945	0.4957	0.5142	0.1603	0.5191	0.6572
		ST	0.3383	0.5154	0.5578	0.2075	0.5500	0.6587
		Ensemble	0.3105	0.5091	0.5403	0.1887	0.5393	0.6611
		SAM3	0.3232	0.5110	0.5392	0.2140	0.5366	0.6192
	Background-Context Fusion	SSL-B	0.3194	0.4882	0.5252	0.1781	0.5276	0.6596
		ST	0.3743	0.5053	0.5789	0.2339	0.5588	0.6602
		Ensemble	0.3385	0.5027	0.5591	0.2056	0.5515	0.6638
		SAM3	0.3615	0.5015	0.5571	0.2429	0.5484	0.6150
RF (Nano)	Base	None	0.2687	0.4846	0.5034	0.1549	0.4770	0.6268
	Standard Fine-tuning	SSL-B	0.2351	0.4331	0.4569	0.1125	0.4426	0.6223
		ST	0.3071	0.4685	0.5266	0.1839	0.4994	0.6267
		Ensemble	0.3005	0.5017	0.5312	0.1778	0.5280	0.6541
		SAM3	0.3136	0.5025	0.5297	0.2037	0.5260	0.6166
	Background-Context Fusion	SSL-B	0.2490	0.4252	0.4595	0.1238	0.4463	0.6226
		ST	0.3365	0.4568	0.5390	0.2024	0.5049	0.6298
		Ensemble	0.3273	0.4942	0.5472	0.1935	0.5404	0.6583
		SAM3	0.3480	0.4927	0.5462	0.2319	0.5378	0.6054
Yolo11 (Large)	Base	None	0.2855	0.4647	0.4926	0.1376	0.4939	0.6402
	Standard Fine-tuning	SSL-B	0.1992	0.3578	0.3199	0.0620	0.3620	0.5724
		ST	0.2932	0.4836	0.4667	0.1748	0.5122	0.6154
		Ensemble	0.2760	0.4729	0.4431	0.1485	0.4972	0.6399
		SAM3	0.3121	0.5043	0.5144	0.2121	0.5372	0.6106
	Background-Context Fusion	SSL-B	0.2011	0.3578	0.3202	0.0618	0.3662	0.5620
		ST	0.2994	0.4710	0.4617	0.1787	0.5104	0.6142
		Ensemble	0.2765	0.4569	0.4315	0.1446	0.4922	0.6263
		SAM3	0.3183	0.4900	0.5061	0.2181	0.5343	0.5922
Yolo11 (Nano)	Base	None	0.1537	0.2763	0.3409	0.0431	0.2628	0.4672
	Standard Fine-tuning	SSL-B	0.1069	0.1969	0.1875	0.0119	0.1703	0.3941
		ST	0.1781	0.3438	0.3486	0.0634	0.3475	0.4947
		Ensemble	0.1925	0.4141	0.3888	0.0820	0.3998	0.5615
		SAM3	0.2066	0.4284	0.4337	0.1211	0.4151	0.5253
	Background-Context Fusion	SSL-B	0.1111	0.2045	0.1934	0.0118	0.1801	0.4131
		ST	0.1870	0.3431	0.3492	0.0728	0.3576	0.4980
		Ensemble	0.1977	0.4026	0.3810	0.0852	0.3990	0.5557
		SAM3	0.2173	0.4167	0.4331	0.1212	0.4170	0.5391

Table 4.4: Comparing Background-Context Fusion (BF) against Standard Fine-tuning (SF) using Wilcoxon signed-rank test on AP (Weighted), computed per scene. Positive median differences indicate BF outperformed SF.

Variant	Labeling Strategy	N Scenes	Median Difference	p-value	Significance
RF-Medium	Ensemble	100	+0.0107	4.49×10^{-8}	***
RF-Medium	SAM3	100	+0.0133	3.13×10^{-6}	***
RF-Medium	SSL-B	100	+0.0089	9.19×10^{-6}	***
RF-Medium	ST	100	+0.0133	1.52×10^{-7}	***
RF-Nano	Ensemble	100	+0.0077	1.15×10^{-5}	***
RF-Nano	SAM3	100	+0.0130	4.41×10^{-5}	***
RF-Nano	SSL-B	100	+0.0049	2.85×10^{-2}	*
RF-Nano	ST	100	+0.0106	1.20×10^{-5}	***
Yolo11-L	Ensemble	100	−0.0043	6.60×10^{-3}	**
Yolo11-L	SAM3	100	−0.0020	3.98×10^{-2}	*
Yolo11-L	SSL-B	100	+0.0001	8.92×10^{-1}	ns
Yolo11-L	ST	100	−0.0019	4.36×10^{-2}	*
Yolo11-N	Ensemble	100	−0.0000	5.57×10^{-1}	ns
Yolo11-N	SAM3	100	+0.0038	3.32×10^{-1}	ns
Yolo11-N	SSL-B	100	+0.0019	1.43×10^{-3}	**
Yolo11-N	ST	100	+0.0071	5.29×10^{-3}	**

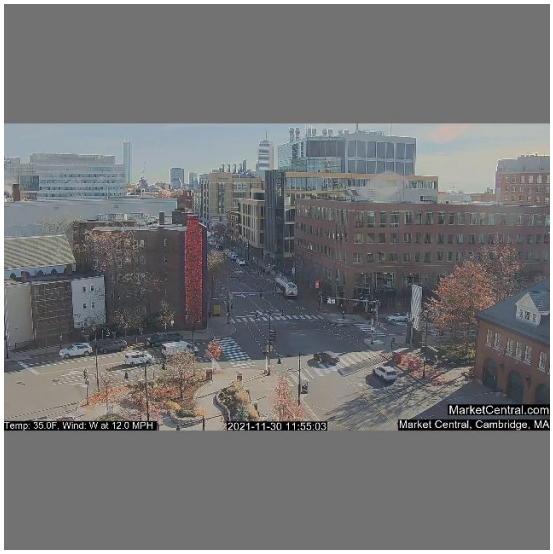
Significance codes: *** $p < 0.001$, ** $p < 0.01$, * $p < 0.05$, ns = not significant.

Median Difference: Positive values indicate BF outperformed SF.

4.4 Seasonality changes

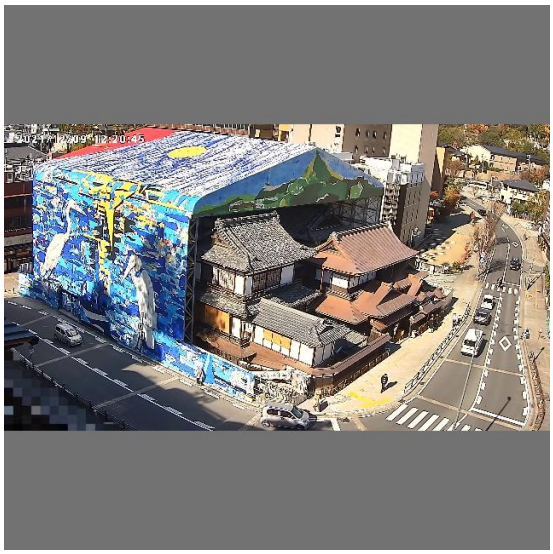
In order to investigate how adaptation results vary under changing environmental and operational conditions, the following section presents the results obtained when models trained on a summer scene are evaluated on the same scene under winter conditions.

Looking at Table 4.5, The models has a better score on the Summer scene compared to the Winter scene, even when looking at the base models. Further, I observe that all configurations that improve AP_{weighted} on Summer scenes fail to retain the same margin over the Base model when evaluated on Winter scenes. In other words, a substantial fraction of the gains achieved in-domain (Summer) do not fully transfer out-of-domain (Winter). For RF-DETR Medium, only the SF with SAM3 variant remains clearly better than the Base model in Winter, whereas all except one variant lose their advantage. In contrast, RF-DETR Nano stands out: all adaptations except the SSL-B variants show positive Δ values relative to the Base in both Summer and Winter, indicating robust generalisation across scene conditions. Among the YOLO11 models, only the YOLO-Nano using SAM3 variants consistently outperform their Base in Winter. The improvements observed in this case are modest relative to the gains achieved in the Summer setting. Given that the summer scene is "harder" according to the base model's performance, and that it also contains more objects



(a) Scene 058 – image

(b) Scene 058 – Object mask



(c) Scene 156 – image

(d) Scene 156 – Object mask

Figure 4.3: Qualitative examples of two scenes where Background-Context Fusion outperforms Standard Fine-tuning. Each row shows the original image (left) and the corresponding object mask image used by the fusion pipeline (right).

(a) Scene 090 – image

(b) Scene 090 – object mask

(c) Scene 125 – image

(d) Scene 125 – object mask

Figure 4.4: Qualitative examples of two scenes where Standard Fine-tuning performs better than Background-Context Fusion. Each row shows the original image (left) and the corresponding object mask image (right).

from the person classes, in the summer setting, I therefore use a relative performance metric.

$$\Delta_{\%} = \frac{AP - AP_{\text{Base}}}{AP_{\text{Base}}} \times 100$$

as a more appropriate basis for comparison. Thus, assuming that we have the $\Delta_{\%}$ for summer and one for winter, I subtract the winter from the summer and if we have a positive value it indicates that winter increase in percentage performance was larger than in summer case.

Across all models and configurations, every model that surpassed the base model in the summer setting exhibits comparatively worse performance in the winter setting (in percentage terms), as reported in the final boldfaced column, they all show negative values.

Table 4.5: Summer → Winter generalisation using AP_{weighted}. All models are trained on Summer data only and evaluated on both Summer and Winter scenes. For each architecture, the Base model is the unadapted reference. We report relative change as $\Delta_{\%} = \frac{AP - AP_{\text{Base}}}{AP_{\text{Base}}} \times 100$ (computed per season). The final column is the Winter − Summer difference in relative change (percentage points).

Model	Model Adaptation	Labeling Strategy	Summer AP	%Δ vs base	Winter AP	%Δ vs base	$\Delta_{\%}$(W−S)
RF (Medium)	Base	*Reference*	0.5061	+0.00%	0.6601	+0.00%	**+0.00**
	Standard Fine-tuning	SSL-B	0.5018	−0.85%	0.6389	−3.21%	**−2.36**
		ST	0.5273	+4.19%	0.6599	−0.03%	**−4.22**
		Ensemble	0.5100	+0.77%	0.6226	−5.68%	**−6.45**
		SAM3	0.5374	+6.18%	0.6726	+1.89%	**−4.29**
	Background-Context Fusion	SSL-B	0.5150	+1.76%	0.6183	−6.33%	**−8.09**
		ST	0.5567	+10.00%	0.6385	−3.27%	**−13.27**
		Ensemble	0.5274	+4.21%	0.6053	−8.30%	**−12.51**
		SAM3	0.5702	+12.67%	0.6512	−1.35%	**−14.01**
RF (Nano)	Base	*Reference*	0.4504	+0.00%	0.5754	+0.00%	**+0.00**
	Standard Fine-tuning	SSL-B	0.4130	−8.30%	0.5364	−6.78%	**+1.53**
		ST	0.5021	+11.48%	0.6238	+8.41%	**−3.07**
		Ensemble	0.5010	+11.23%	0.6072	+5.53%	**−5.71**
		SAM3	0.5291	+17.47%	0.6564	+14.08%	**−3.40**
	Background-Context Fusion	SSL-B	0.4186	−7.06%	0.5239	−8.95%	**−1.89**
		ST	0.5217	+15.83%	0.5962	+3.61%	**−12.22**
		Ensemble	0.5123	+13.74%	0.5885	+2.28%	**−11.47**
		SAM3	0.5522	+22.60%	0.6228	+8.24%	**−14.36**
Yolo11 (Large)	Base	*Reference*	0.5271	+0.00%	0.6760	+0.00%	**+0.00**
	Standard Fine-tuning	SSL-B	0.4394	−16.64%	0.5919	−12.44%	**+4.20**
		ST	0.5711	+8.35%	0.6490	−3.99%	**−12.34**
		Ensemble	0.5192	−1.50%	0.6065	−10.28%	**−8.78**
		SAM3	0.5909	+12.10%	0.6689	−1.05%	**−13.15**
	Background-Context Fusion	SSL-B	0.4270	−18.99%	0.5572	−17.57%	**+1.42**
		ST	0.5784	+9.73%	0.6263	−7.35%	**−17.08**
		Ensemble	0.5146	−2.37%	0.5836	−13.67%	**−11.30**
		SAM3	0.5980	+13.45%	0.6412	−5.15%	**−18.60**
Yolo11 (Nano)	Base	*Reference*	0.2993	+0.00%	0.4531	+0.00%	**+0.00**
	Standard Fine-tuning	SSL-B	0.2314	−22.69%	0.3330	−26.51%	**−3.82**
		ST	0.3409	+13.90%	0.4360	−3.77%	**−17.67**
		Ensemble	0.3888	+29.90%	0.4343	−4.15%	**−34.05**
		SAM3	0.4099	+36.95%	0.4641	+2.43%	**−34.53**
	Background-Context Fusion	SSL-B	0.2274	−24.02%	0.3134	−30.83%	**−6.81**
		ST	0.3448	+15.20%	0.4390	−3.11%	**−18.31**
		Ensemble	0.3836	+28.17%	0.4349	−4.02%	**−32.18**
		SAM3	0.4116	+37.52%	0.4571	+0.88%	**−36.64**

5

Conclusion and Discussion

5.1 Discussion

This section presents a discussion specifically aimed at addressing the research questions. Specifically, we examine how YOLO11(CNN) and RF-DETR (Transformer) compare in adaptation capability within static scenes, to what extent self-supervised scene adaptation can enable smaller models compared to larger ones, and whether a potential on-device strategy for self-supervised learning can achieve performance parity with more resource-heavy methods. I also assess whether the integration of background extraction and feature fusion in a fixed-camera environment provides a performance improvement for the chosen models, and quantify how large the accuracy drop is under seasonal and weather shifts for an adapted model.

5.1.1 Adaptation Capabilities

In this section, I will cover RQ1: **How do YOLO11(CNN) and RF-DETR (Transformer) compare in adaptation capability within static scenes?**

The RF-DETR model consistently exhibits superior performance across the 100Scenes dataset, additionally exhibiting faster and more stable convergence behavior compared to YOLO11.A comparable convergence behaviour for RF-DETR was reported by Sapkota et al. [17], as discussed in Section 2.6. In their work, RF-DETR exhibited rapid and stable convergence despite employing an unfrozen backbone, in contrast to the frozen-backbone configuration adopted in the present study. This suggests that the lower number of % trainable-parameters for the RF-DETR (22.6% and 29.2%) architecture compared to the YOLO11-models (47.3% and 49.3%) is not the crucial factor for faster convergence.

The specific allocation and organization of parameters within the architecture likely play a pivotal role in determining its capacity for generalisation. RF-DETR relies

on a "heavy backbone" design, utilizing the massive pre-trained DiNOv2 backbone [18]. In this configuration, the vast majority of the model's capacity resides in the frozen backbone, which encapsulates robust general-purpose semantic representations learned from vast amounts of data. In contrast, the YOLO11 architectures examined are designed with a heavier neck and head. In this configuration, just under 50% of the parameters are located outside the backbone compared to RF-DETRS (22.6% (nano), 29.2%) less trainable parameters. While this allows the model to adapt more specifically to the dataset, it increases the risk of overfitting. This expectation is reflected in the results. As presented in the Performance Delta vs. Base (Table 4.2), the YOLO11 models are more negatively impacted by inadequate labeling strategies, resulting in a more pronounced degradation in performance. Concurrently, when comparing larger and smaller model variants, the relative performance gains achieved under enhanced training conditions are more pronounced for the YOLO11 architectures than for their counterparts within the RF-DETR models. For example, when high-quality annotations generated by SAM3 were employed, YOLO11-Large exhibited a greater relative performance improvement over its corresponding base model than RF-DETR-Medium did over its own base model. Thus, if trained with high-quality labels and appropriate optimization strategies on a given scene, YOLO11 models may offer greater potential performance gains, albeit with an increased risk of overfitting even with a frozen backbone. This is further highlighted in the seasonality change results for scene 001. The relative change in performance is consistently worse when looking at the YOLO11 model configurations than RF-DETR model configurations. For instance, the RF-Medium using SAM3 labels and standard fine-tuning was -4.29 percentage points worse in the winter scene, while YOLO11-Large with the same configurations showed a greater degradation of performance that was 3 times large, being -13.15 percentage points worse in the winter scene. Thus, it further proves that the robustness of the RF-DETR architecture is better as we can see better results across Scenes100 and the sesonality change results.

Convergence smoothness is further governed by the loss landscape defined by label assignment strategies. While YOLO11's heuristic assignment induces gradient variance through local'many-to-one" ambiguity, RF-DETR, using a global bipartite matching and the incorporation of Group-DETR described in Section 2.4.2[52], provides dense gradient flow through auxiliary query groups, effectively stabilizing the training dynamics. However, it comes with some minor computational costs.

The observed differences in generalization performance can also be attributed to the distinct architectural properties of CNNs and Transformer models, in particular, to the mechanisms by which they extract, represent, and selectively attend to features. YOLO11 is still largely shaped by the CNN inductive bias of locality [35], it builds features by stacking convolutional layers, where each layer aggregates information from a local neighborhood. Global context is therefore reached only indirectly, by gradually expanding the effective receptive field across many layers (Section 2.3.1). Although YOLO11 has evolved (Section 2.3.2) and even includes attention-

components, it still heavily relies on locality. The model has three more or less decoupled feature-maps, which it relies on for final predictions. The decoupling from each other, in one sense, could make it more robust, but it may miss fine-overlapping patterns.

In contrast, the RF-DETR architecture uses attention across all parts and heavily relies on it as the primary mechanism for information aggregation, effectively focusing on more important "areas". Its design includes efficient attention variants such as Deformable Attention and Mixed-Query Selection (Section 2.4.2) [14], [16], [48]. Instead of depending on fixed, local feature extraction, Transformers prioritize information based on relative importance, they can directly focus on the most relevant regions and suppress less relevant background. Moreover, in the decoder, queries act like detection slots, which are derived from the same feature representation and operate in the same embedding space. This means that they effectively compete for representation capacity, which can encourage a more consistent and globally coordinated interpretation of the scene. In theory, this global coordination and attention focus should make the model less sensitive to local divergence and therefore more robust when conditions shift.

However, it is also important to note that RF-DETR entails higher hardware requirements, both during training and inference, in terms of memory consumption and the need for hardware capable of extensive parallelization to achieve the inference speeds reported in prior work [64]. Thus, one should not choose RF-DETR solely based on performance, as YOLO11 still leads when it comes to parameter-efficient needs, but RF-DETR nano is extremely close to being as efficient as YOLO11-large.

RF-DETR's superior stability and generalization arise from the combined effects of its heavily pretrained backbone, efficient architectural attention mechanisms, and globally consistent bipartite-matching loss. These design elements allow RF-DETR to function as a generalist detector that remains robust under label noise and domain shift but comes with an increase in hardware requirements. By contrast, YOLO11 architectures place a larger share of parameters in the neck and head, making them highly adaptable and capable of strong peak performance on well-labeled, scene-specific datasets in this configuration. However, this same flexibility also makes YOLO11 more sensitive to imperfect supervision and more prone to overfitting.

5.1.2 Smaller models compared to larger

This section focuses on RQ2 to further understand how a smaller model can compete with a larger one. **To what extent can self-supervised scene adaptation enable smaller models compared to larger ones?**

The results show that there is a tendency for a pattern between model size and the efficacy of self-supervised adaptation in the YOLO11-models while it being less clear in the RF-DETR models. As evidenced in Table 4.2, using the label-strategy of ST, the YOLO11-Nano increases by as much as 6.22% units from the Base in AP_{weighted}, while YOLO11-Large only yields an increase of up to 2.17%units. In the RF-DETR models, there are no such patterns, as the medium size achieves 3.77% and the Nano 3.61%. However, looking at the results when using the heavier methods, Ensemble and SAM3, which deploy larger models. There is a general pattern across the architectures regarding model size. The smaller variants compared to the larger variants exhibit much greater increases in performance. YOLO11-Nano performance using SAM3 increases up to 13.45%, while YOLO11-Large in the same configuration yields an increase of 4.94%. RF-DETR models show the same pattern as the RF-DETR-Nano using SAM3, which increases up to 6.46%, while the RF-DETR-Medium with the same configuration yields a 3.48% increase. Showing that there is a greater performance to achieve. Allowing YOLO11-Nano (Best: 0.3750 AP) to approach the performance tier of the unadapted Base YOLO11-Large (Base: 0.4227 AP). These methods would theoretically not be purely self-supervised but would involve knowledge distillation, as they are being trained by larger models. Additionally, as reported in the scene-specific performance analysis in Section 4.3 on the adaptation strategy, YOLO11-Nano with background-context fusion (BF) achieved a weighted average precision ($AP_{weighted}$) of 0.5123 when combined with SAM3 and 0.4676 when trained using purely self-supervised ST, whereas the YOLO11-Large base model achieved 0.5202. These results demonstrate that a model with approximately 10× fewer parameters can match the performance of a non-specialized, substantially larger model. This result indicates that a compact model, when appropriately configured and trained, can outperform substantially larger models on the same task. It further suggests that smaller, specialized models can be competitive with, or superior to, larger general-purpose models on highly specialized tasks.

Moreover, the current methodology of self-supervised learning (referring to labeling strategies SSL-B and ST) employs only a single round of adaptation: the base model first generates pseudo-labels, and the target model is then trained on these labels. Since the adapted model now substantially outperforms the base model used to produce the initial labels (in the ST setting), an additional iteration of adaptation—where the improved model is used to regenerate labels—could plausibly yield further performance gains.

5.1.3 SAHI + ByteTrack Yields performance increase at top level

This section covers the RQ3, **Can a potential on-device strategy for self-supervised learning achieve performance in parity with more resource-heavy methods?**

The results unequivocally demonstrate that the proposed SAHI + ByteTrack (ST) strategy is highly effective, often competing with or superseding computationally heavier methods like the Ensemble approach. While the impact is slightly more pronounced in the larger models compared to the heavier methods. This is likely due to their closer absolute performance, meaning that the base model's performance is closer to that of the models used in the heavier methods (SAM3, Ensemble). Furthermore, as discussed in Section 5.1.2, the smaller model benefits more from the distilled knowledge provided by the heavier methods.

The most profound insight from these experiments lies in the nature of the improvements. The substantial gains observed in small and medium objects are shown in the results in Table 4.3. RF-DETR-Nano, with the use of ST, enhances performance in relation to the base model from 0.15 $\Rightarrow$ 0.20 for small objects and 0.48 $\Rightarrow$ 0.51 for medium objects, while Large objects increase by less than 0.04 points. This suggests that the decisive factor in successful adaptation is not merely the size of the teacher model, but the strategic recovery of "hard negatives." The "hard negatives" refer to what is covered in the theory in Section 2.5: false negatives, which are detections that are missed. Thus, they constitute objects that are inherently difficult for detectors to identify[56]. The proposed ST method excels precisely because it targets the specific weaknesses of standard detectors. SAHI [60] covered in the theory (Section 2.5) enables small objects to be missed by exploiting high-resolution slicing. Simultaneously, the bi-directional implementation of ByteTrack (Section 2.5) recovers occluded or low-confidence instances—candidates that are typically discarded by standard thresholds but are crucial for robust learning. This further confirms the previous results that are covered in Section 2.5[56] the recovery of 'hard negatives' boosts performance [21], [57] .

While the commercial-grade SAM3 model yields excellent results due to consistent tracking and high-resolution processing, its reliance on unidirectional tracking limits its ability to recover past occlusions compared to the bi-directional approach used in ST. This reinforces the conclusion that superior adaptation is driven by how effectively a strategy complements a model's inherent limitations rather than by brute-force scaling. By shifting the focus toward these hard negatives—specifically the small, occluded, and low-confidence examples—future adaptation strategies can push performance boundaries further, potentially achieving higher accuracy with less data by concentrating specifically on the most information-rich examples.

5.1.4 The Background Context Fusion impact on models

This section focuses on the Background Context Fusion (BF) and attempts to answer RQ4 **Does the integration of background extraction and feature fusion in a fixed-camera environment provide a performance improvement for the chosen models?**

The experimental results indicate a clear architectural divergence in the effectiveness of Background-Context Fusion (BF). Specifically, both RF-DETR variants exhibit a consistent performance improvement when BF is employed, whereas its impact on the YOLO11 models is inconclusive or negligible. Moreover, the best-performing configurations of RF-DETR rely on BF rather than Standard Finetuning (SF), while, conversely, the optimal YOLO11 configurations are obtained using SF. The Wilcoxon signed-rank test reported in Section 4.3 further corroborates this pattern, it reveals a statistically significant positive effect in favor of BF over SF across all RF-DETR configurations, whereas, for YOLO11, the relative benefit of BF versus SF is contingent upon the specific configuration. I would argue that a main reason for this disparity is how information is processed in the different architectures. For clarity, I will explain a simplified example. Using Background Context Fusion, it provides features based on both the original image and the object mask separately (background being gray (0.5) separately, while objects should diverge from the background), and they are then element-wise averaged. Backbones are solely pretrained on regular images with no object masks. three situations can occur: either we are in a position of an object, in a position of a background, or both. Thus, when the backbones examine a pixel of an object in the image, they receive a clear signal from the original-image, while from the object mask they receive a diluted signal, and these features are later averaged. Effectively, this "dilutes" the feature intensity. Further, for instance, a strong edge value of 1.0 in the image in a background position will be diluted to 0.5 (assuming no feature from object mask input). For the CNNs driving YOLO11, this reduction in contrast degrades feature fidelity, "washing out" the local texture cues essential for detection. In contrast, RF-DETR is more robust to this dilution due to attention mechanism, even with diluted pixels, the mechanism relies on attention scores, allowing the Softmax function to amplify the signal of relevant objects over the background, so it should still deliver higher-fidelity features. Similar behavior was reported in Section 2.5, where RF-DETR, in comparison to YOLO12 (CNN), exhibited superior performance under conditions of object occlusion [67]. In the present study, this behavior can be interpreted as the result of an imperfect fusion map that effectively induces a form of “synthetic occlusion,” through which the Transformer is able to infer relevant information by exploiting its attention mechanism via attention scores. Furthermore, RF-DETR’s Mixed-Query Selection strategy, described in Section 2.4.2, allows the object-queries to be initialized in the most promising areas of the image. This function is similar to Region of Interest (RoI) proposals used in Faster R-CNN, which being the architecture that increased performance through background feature fusion in previous literature, as described in Section 2.5[21]. This enables the model to spatially filter the input and incorporate background context solely at the key object locations. Conversely, YOLO11’s dense, grid-based architecture forces the processing of the noisy fusion signal across the entire spatial domain. While extended training horizons might allow YOLO11 to adapt to this distribution, the immediate benefit is very scene-dependent.

Another similarity between Faster R-CNN (used in previous successful performance increase) and the RF-DETR architecture is the use of a computationally heavy

backbone [21]. It is possible that the lighter YOLO11-based backbones do not yield sufficiently rich or stable feature representations when diluted or disturbed by the object mask fusion (including background and objects), thereby hindering effective feature disentanglement.

Consequently, the architectural design of RF-DETR renders it substantially more suitable for background-extraction-based fusion. However, achieving a net performance gain still necessitates careful adaptation to the specific topological characteristics of the scene.

However, the qualitative analysis covered in the results, Section 4.3, reveals a clear pattern. Certain scenes, when incorporating BF, consistently boost performance, regardless of the architecture. Complex scenes rich in structural edges seem to add a lot of noise, stealing attention and making the BF perform much better. For instance, consider the scenes in 058 and 156, one shows a traffic intersection in a large city, while the other depicts construction occupying a large portion of the image, with a road running alongside it. In contrast, exemplified by scenes 090 and 125 in Section 4.3, scenes dominated by blank white backgrounds or heavy occlusion, where objects are present (as shown in the results of the Christmas decoration lights scene 125), inject harmful noise that drags performance down, making Standard Fine-Tuning (SF) the decisively better choice in those settings.

A potential reason why a white background might be worse in the winter scenarios is that the original input almost works as a "natural mask" due to the extreme contrast of dark objects against white snow, yielding strong feature activations from pretrained backbones. Conversely, the generated object mask suffers from severe texture collapse, reducing identifiable vehicles to "black blobs", which can be seen in the winter scene previously discussed. A closer inspection of the object masks in the winter scenes reveals pronounced artifacts, which are likely attributable to the high reflectance sensitivity of the snow-covered surfaces rather than to the underlying semantic structure.. Since the backbone is strictly frozen, it lacks the capacity to adapt to these featureless blobs or suppress the observed artifacts, resulting in a secondary branch output dominated by noise or null activations. Consequently, as discussed in the previous paragraph, the element-wise averaging strategy acts to destroy the good features from the original image by diluting the sharp, high-confidence features of the original image with the weak responses of the mask branch. Thus, I argue that the dilution becomes even greater in this setting, and as the backbone is frozen and cannot adapt, it is hard for the rest of the network to decode the low-quality features.

5.1.5 Seasonality changes in scenes

The final section covers and discusses the final RQ5, **How does the accuracy of the adapted model under seasonal shifts compare to the performance of the non-adapted base model and its trained-domain performance?**

A consistent pattern emerges in which all models exhibit a decrease in relative performance compared with the base model. However, it must be taken into account that the Winter scene exhibits consistently higher scores across all models compared to the Summer scene; consequently, the relative increase may be more pronounced in the Winter scene. Surprisingly, the background-context fusion strategy does not enhance model robustness, instead, it degrades performance. One possible explanation is that this strategy induces a form of overfitting to the background information. Specifically, the average "background" characteristics present within the object-mask features are likely to differ from the average "background" features extracted from the full images in the new context. This discrepancy induces a distributional shift that degrades generalization performance, as also discussed in the previous section.

Moreover, the results strengthen the indications that YOLO11 is more sensitive to these contextual and seasonal variations, whereas RF-DETR exhibits comparatively greater robustness.

The selection of an appropriate deployment strategy ultimately remains contingent upon the specific hardware configuration and the context of the scene. Ideally, where feasible, an optimised ST pipeline that continuously adapts to a dynamically changing environment would be preferred, and the choice of model would be primarily based on the available hardware. At present, this concept is primarily theoretical, and additional experiments are required to translate it into a practical deployment setting. Nevertheless, the findings of this study demonstrate the potential of an on-device learning strategy, warranting further investigation.

5.2 Recommendation for future works

The scenes have only been evaluated once per model configuration thus, the same seed is being used. For a more robust result, fine-tuning and evaluating more than once per scene would be ideal. This approach would also facilitate a more rigorous statistical analysis of the data and support the derivation of more definitive and transparent conclusions.

Furthermore, hardware analysis and requirements have not been central, and assuming that previous literature results stand. However, this could be misleading, as FP16

was used for RF-DETR latency measures while accuracy was assessed using FP32. Additionally, the latencies are not consistent in the literature. The commonly reported latency and performance metrics are given for varying resolutions, which further complicates the generalization of previous literature and its comparison with the results of this study. Additionally, a test in a real-deployment environment would be needed to confirm the trade-offs for memory and latency and to further understand the benefits of each model.

Another interesting direction for future work, as previously touched upon in the labeling strategy discussion, involves the iterative refinement of pseudo-labels. The current self-supervised methodology is limited to a single iteration of adaptation: the base model generates the initial pseudo-labels, and the target model is trained on them. Since the adapted model now significantly outperforms the base model used to produce those initial labels (particularly in the ST setting), utilizing this improved model to regenerate the pseudo-labels for a second round of training could plausibly yield further performance gains. This iterative "bootstrapping" process would allow the model to progressively correct early labeling errors and refine its own supervision signal.

Future research should prioritize on-device testing of the proposed SAHI+ByteTrack pseudo-labeling strategy to assess its real-world applicability. This includes evaluating hardware requirements and identifying necessary optimizations for edge deployment. Furthermore, it is crucial to investigate the data efficiency of the adaptation process. As discussed previously, significant performance gains often stem from learning specific "hard negatives," suggesting that model improvement might saturate with relatively small datasets. Determining the minimal data volume required for this saturation would be valuable; if only a small amount of high-quality data is needed, rapid on-device re-training could become an even more advantageous solution for adapting to changing environments.

To further add to a potential strategy when deploying and training a model. It would be very interesting to compare a domain-trained model across the different configurations. By this, I mean that instead of training for a specific scene, the various model configurations used in this study would be trained on 80 of the scenes and tested on 20 for validation; one can apply cross-validation for even more robust conclusions. With those results, one can compare the different strategies. Would it be better to-retrain a model for the scene, making it more sensitive to changes and at risk of degrading performance, or is it better to have a general model trained for the domain, which carries less risk of degrading performance but might not reach the peaks of a scene-specific version?

The Background-Context Fusion, in its current implementation, is more hardware-

intensive and increases latency, a concern also noted in previous literature. However, I would like to point out the optimization that was done in Group-DETR, allowing for barely any increase in memory or latency. thus, similar to the Group-DETR one can run the backbone-passes in parallel, the increase in hardware requirements would be minimal, and latency would be slightly worse but potentially minimal. This assumes that one has hardware that can run in parallel and enough memory. However, this needs to be robustly tested to be confirmed, especially in an edge-environment, as this strict environment would most likely also lead to an increase in latency.

The resolution has been fixed to 640×640. For a different resolution, it is not evident that it would lead to the same conclusions, although I would argue that the core conclusion is still applicable.

D-Fine has demonstrated state-of-the-art performance by formulating the task as iterative refinement detection [23], making it another noteworthy architecture with which to conduct more comparative research. Nonetheless, I will not discuss this method here due to time and resource limitations, instead focusing on more established architectures.

Hyperparameter tuning has not been considered, and the use of general parameters, if not otherwise specified, has been implemented. Better performance could have been achieved if this had been taken into account.

5.3 Industry Recommendations

When it comes to architecture selection for industrial deployment, the choice of architecture must be dictated by the specific constraints of the environment and hardware. RF-DETR is the superior candidate for performance and robustness. Conversely, YOLO11 is a good choice for very resource-constrained edge devices, where its high parameter efficiency relative to performance is beneficial, provided the environment remains relatively stable, as overfitting becomes a real risk if it is trained on a specific scene.

For pseudo-labeling strategies, "hard-negative" should be the focus, and brute-force data scaling should be abandoned for the adaptation of models. SAHI + ByteTrack is a great choice for a hardware constrained environment, while SAM3 creates robust labels for all types of objects and provides a good-quality labeling strategy if one has the hardware to run the model. However, it is important to focus on effectively recovering small, occluded, and low-confidence instances that are typically missed by standard detectors. Future deployment pipelines should prioritize these "hard" examples over sheer data volume to maximize efficiency.

Specific scenes require great caution when implementing Background-Context Fusion (BF). While the technique is architecturally better suited to RF-DETR, its success is predominantly scene-dependent. Thus, an understanding of the scene and how one's detector processes the data is key for a successful implementation.

While large, general-purpose models inherently offer greater robustness and reduced risk of overfitting, the standard practice of defaulting to the largest possible model should be reconsidered for fixed-camera surveillance. The results demonstrate that the efficacy of scene-specific fine-tuning allows a highly specialized small model to match the accuracy of a larger general-purpose model. However, this may potentially lead to a higher risk of overfitting and sensitivity to changes. The deployment strategy should focus on finding the correct balance of size, specialization, and risk that aligns with the specific deployment environment. Consequently, smaller, specialized models often offer a far more cost-effective solution that achieves high-performance detection without the computational overhead of generalist architectures.

5.4 Conclusions

The aim of this thesis was to evaluate the performance of two state-of-the-art real-time object-detecting architectures in the context of surveillance for a fixed-camera scene. RF-DETR consistently achieved higher accuracy, faster and smoother convergence, and greater robustness than YOLO11 variants. However, it requires more powerful hardware. In contrast, with a frozen backbone, YOLO11's larger proportion of trainable capacity in the neck/head enables scene-specific adaptability and higher relative gains under high-quality labeling. This, on the other hand, also increases sensitivity to imperfect pseudo-labels and the risk of overfitting. It is shown that a small but highly specialized model with scene-adaptation and higher quality training labels is able to compete with a larger general model. The proposed on-device adaptation, SAHI + bi-directional ByteTrack provided reliable performance gains across architectures and sizes, largely by recovering hard negatives (small, occluded, low-confidence). Background-Context Fusion consistently improved RF-DETR across 100Scenes but was inconsistent for YOLO11 and did not increase seasonality robustness, suggesting that BF can induce background-dependent overfitting under domain shift. Through qualitative analysis, certain scenes can benefit no matter the architectures, thus highlighting that the nature of the scene is crucial for performance. All adapted models exhibit a decrease in relative performance compared with the non-adapted models during a seasonal domain shift. Thus, self-supervised scene adaptation in fixed-camera surveillance is fully viable for enhancing performance, provided that the auto-annotation methodology is carefully selected and that inherent variations between scenes are taken into account.

Bibliography

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, F. Pereira, C. Burges, L. Bottou, and K. Weinberger, Eds., vol. 25, Curran Associates, Inc., 2012. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2012/file/c399862d3b9d6b76c8436e924a6
Paper.pdf.

[2] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, *You only look once: Unified, real-time object detection*, 2016. arXiv: 1506.02640 [cs.CV]. [Online]. Available: https://arxiv.org/abs/1506.02640.

[3] X. Huang et al., *Pp-yolov2: A practical object detector*, 2021. arXiv: 2104.10419 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2104.10419.

[4] J. Redmon and A. Farhadi, *Yolov3: An incremental improvement*, 2018. arXiv: 1804.02767 [cs.CV]. [Online]. Available: https://arxiv.org/abs/1804.02767.

[5] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, *Yolov4: Optimal speed and accuracy of object detection*, 2020. arXiv: 2004.10934 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2004.10934.

[6] R. Khanam and M. Hussain, *What is yolov5: A deep look into the internal features of the popular object detector*, 2024. arXiv: 2407.20892 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2407.20892.

[7] C. Li et al., *Yolov6: A single-stage object detection framework for industrial applications*, 2022. arXiv: 2209.02976 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2209.02976.

[8] C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao, *Yolov7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors*, 2022. arXiv: 2207.02696 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2207.02696.

[9] M. Yaseen, *What is yolov8: An in-depth exploration of the internal features of the next-generation object detector*, 2024. arXiv: 2408.15857 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2408.15857.

[10] C.-Y. Wang, I.-H. Yeh, and H.-Y. M. Liao, *Yolov9: Learning what you want to learn using programmable gradient information*, 2024. arXiv: 2402.13616 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2402.13616.

[11] A. Wang et al., *Yolov10: Real-time end-to-end object detection*, 2024. arXiv: 2405.14458 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2405.14458.

[12] R. Khanam and M. Hussain, *Yolov11: An overview of the key architectural enhancements*, 2024. arXiv: 2410.17725 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2410.17725.

[13] A. Vaswani et al., *Attention is all you need*, 2023. arXiv: 1706.03762 [cs.CL]. [Online]. Available: https://arxiv.org/abs/1706.03762.

[14] A. Dosovitskiy et al., *An image is worth 16x16 words: Transformers for image recognition at scale*, 2021. arXiv: 2010.11929 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2010.11929.

[15] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, *End-to-end object detection with transformers*, 2020. arXiv: 2005.12872 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2005.12872.

[16] Q. Chen et al., *Lw-detr: A transformer replacement to yolo for real-time detection*, 2024. arXiv: 2406.03459 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2406.03459.

[17] R. Sapkota, R. H. Cheppally, A. Sharda, and M. Karkee, *Rf-detr object detection vs yolov12 : A study of transformer-based and cnn-based architectures for single-class and multi-class greenfruit detection in complex orchard environments under label ambiguity*, 2025. arXiv: 2504.13099 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2504.13099.

[18] M. Oquab et al., *Dinov2: Learning robust visual features without supervision*, 2024. arXiv: 2304.07193 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2304.07193.

[19] T.-Y. Lin et al., *Microsoft coco: Common objects in context*, 2015. arXiv: 1405.0312 [cs.CV]. [Online]. Available: https://arxiv.org/abs/1405.0312.

[20] P. Oza, V. A. Sindagi, V. VS, and V. M. Patel, *Unsupervised domain adaptation of object detectors: A survey*, 2021. arXiv: 2105.13502 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2105.13502.

[21] Z. Zhang and M. Hoai, "Object detection with self-supervised scene adaptation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2023, pp. 21 589–21 599.

[22] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in Neural Information Processing Systems (NIPS)*, 2015, pp. 91–99.

[23] Y. Peng, H. Li, P. Wu, Y. Zhang, X. Sun, and F. Wu, *D-fine: Redefine regression task in detrs as fine-grained distribution refinement*, 2024. arXiv: 2410.13842 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2410.13842.

[24] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International journal of computer vision*, vol. 60, no. 2, pp. 91–110, 2004.

[25] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, vol. 1, 2005, 886–893 vol. 1. DOI: 10.1109/CVPR.2005.177.

[26] M. L. Minsky and S. Papert, *Perceptrons: An Essay on Computational Geometry*. Cambridge, MA: MIT Press, 1969.

[27] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Networks*, vol. 2, no. 5, pp. 359–366, 1989. DOI: 10.1016/0893-6080(89)90020-8.

[28] G. Cybenko, "Approximation by superpositions of a sigmoidal function," *Mathematics of Control, Signals and Systems*, vol. 2, no. 4, pp. 303–314, 1989. DOI: 10.1007/BF02551274.

[29] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation," *Nature*, vol. 323, no. 6088, pp. 533–536, 1986. DOI: 10.1038/323533a0.

[30] F. Rosenblatt, *Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms*. Washington, DC, USA: Spartan Books, 1962.

[31] I. Loshchilov and F. Hutter, *Decoupled weight decay regularization*, 2019. arXiv: 1711.05101 [cs.LG]. [Online]. Available: https://arxiv.org/abs/1711.05101.

[32] D. P. Kingma and J. Ba, *Adam: A method for stochastic optimization*, 2017. arXiv: 1412.6980 [cs.LG]. [Online]. Available: https://arxiv.org/abs/1412.6980.

[33] D. H. Hubel and T. N. Wiesel, "Receptive fields of single neurones in the cat's striate cortex," *Journal of Physiology*, vol. 148, pp. 574–591, 1959.

[34] K. Fukushima, "Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position," *Biological Cybernetics*, vol. 36, pp. 193–202, 1980.

[35] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998. DOI: 10.1109/5.726791.

[36] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Neuroips*, pp. 1097–1105, 2012.

[37] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014, pp. 580–587.

[38] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial pyramid pooling in deep convolutional networks for visual recognition," in *Computer Vision – ECCV 2014*. Springer International Publishing, 2014, pp. 346–361, ISBN: 9783319105789. DOI: 10.1007/978-3-319-10578-9_23. [Online]. Available: http://dx.doi.org/10.1007/978-3-319-10578-9_23.

[39] R. Girshick, "Fast r-cnn," in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2015, pp. 1440–1448.

[40] J. Redmon and A. Farhadi, "Yolo9000: Better, faster, stronger," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 7263–7271.

[41] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, *Feature pyramid networks for object detection*, 2017. arXiv: 1612.03144 [cs.CV]. [Online]. Available: https://arxiv.org/abs/1612.03144.

[42] Z. Ge, S. Liu, F. Wang, Z. Li, and J. Sun, *Yolox: Exceeding yolo series in 2021*, 2021. arXiv: 2107.08430 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2107.08430.

[43] C.-Y. Wang, H.-Y. M. Liao, Y.-H. Wu, P.-Y. Chen, J.-W. Hsieh, and I.-H. Yeh, "Cspnet: A new backbone that can enhance learning capability of cnn," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2020, pp. 390–391.

[44] S. Fang, Y. Shen, H. Zou, Y. Yin, W. Jin, and H. Zhou, "Birds-yolo: A bird detection model for dongting lake based on modified yolov11," *Biology*, vol. 14, no. 11, 2025, ISSN: 2079-7737. DOI: 10.3390/biology14111515. [Online]. Available: https://www.mdpi.com/2079-7737/14/11/1515.

[45] G. Jocher and J. Qiu, *Ultralytics yolo11*, version 11.0.0, 2024. [Online]. Available: https://github.com/ultralytics/ultralytics.

[46] A. Vaswani et al., "Attention is all you need," in *Advances in neural information processing systems (NIPS)*, 2017, pp. 5998–6008.

[47] I. Robinson, P. Robicheaux, M. Popov, D. Ramanan, and N. Peri, *Rf-detr: Neural architecture search for real-time detection transformers*, 2025. arXiv: 2511.09554 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2511.09554.

[48] X. Zhu, W. Su, L. Lu, B. Li, X. Wang, and J. Dai, *Deformable detr: Deformable transformers for end-to-end object detection*, 2021. arXiv: 2010.04159 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2010.04159.

[49] Y. Zhao et al., *Detrs beat yolos on real-time object detection*, 2023. arXiv: 2304.08069 [cs.CV].

[50] Y. Li, H. Mao, R. Girshick, and K. He, *Exploring plain vision transformer backbones for object detection*, 2022. arXiv: 2203.16527 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2203.16527.

[51] H. Zhang et al., *Dino: Detr with improved denoising anchor boxes for end-to-end object detection*, 2022. arXiv: 2203.03605 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2203.03605.

[52] Q. Chen et al., "Group detr: Fast detr training with group-wise one-to-many assignment," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023, pp. 6610–6619.

[53] H. Rezatofighi, N. Tsoi, J. Gwak, A. Sadeghian, I. Reid, and S. Savarese, *Generalized intersection over union: A metric and a loss for bounding box regression*, 2019. arXiv: 1902.09630 [cs.CV]. [Online]. Available: https://arxiv.org/abs/1902.09630.

[54] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick, *Momentum contrast for unsupervised visual representation learning*, 2020. arXiv: 1911.05722 [cs.CV]. [Online]. Available: https://arxiv.org/abs/1911.05722.

[55] K. He, X. Chen, S. Xie, Y. Li, P. Dollár, and R. Girshick, "Masked autoencoders are scalable vision learners," in *CVPR*, 2022. [Online]. Available: https://openaccess.thecvf.com/content/CVPR2022/html/He_Masked_Autoencoders_Are_Scalable_Vision_Learners_CVPR_2022_paper.html.

[56] A. RoyChowdhury et al., *Automatic adaptation of object detectors to new domains using self-training*, 2019. arXiv: 1904.07305 [cs.CV]. [Online]. Available: https://arxiv.org/abs/1904.07305.

[57] S. Sajid, Z. Aziz, O. Urmonov, and H. Kim, "Improving object detection accuracy with self-training based on bi-directional pseudo label recovery," *Electronics*, vol. 13, no. 12, 2024, ISSN: 2079-9292. [Online]. Available: https://www.mdpi.com/2079-9292/13/12/2230.

[58] Y. Zhang et al., *Bytetrack: Multi-object tracking by associating every detection box*, 2022. arXiv: 2110.06864 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2110.06864.

[59] M. Jamali, P. Davidsson, R. Khoshkangini, M. G. Ljungqvist, and R. Mihailescu, "Specialized indoor and outdoor scene-specific object detection models," in *Sixteenth International Conference on Machine Vision (ICMV 2023)*, ser. Proceedings of SPIE, 2024. DOI: 10.1117/12.3023479. [Online]. Available: https://mau.diva-portal.org/smash/get/diva2:1846539/FULLTEXT01.pdf.

[60] F. C. Akyon, S. Onur Altinuc, and A. Temizel, "Slicing aided hyper inference and fine-tuning for small object detection," in *2022 IEEE International Conference on Image Processing (ICIP)*, IEEE, Oct. 2022, pp. 966–970. DOI: 10.1109/icip46576.2022.9897990. [Online]. Available: http://dx.doi.org/10.1109/ICIP46576.2022.9897990.

[61] G. Bhat, M. Danelljan, L. Van Gool, and R. Timofte, "Learning discriminative model prediction for tracking," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019, pp. 6182–6191.

[62] N. Carion et al., *Sam 3: Segment anything with concepts*, 2025. arXiv: 2511.16719 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2511.16719.

[63] G. Jocher, A. Chaurasia, and J. Qiu, *Ultralytics yolo*, version 8.0.0, Accessed: 2025-12-13, 2025. [Online]. Available: https://github.com/ultralytics/ultralytics.

[64] I. Robinson, P. Robicheaux, and M. Popov, *Rf-detr: Sota real-time object detection model*, Accessed: 2025-12-13, 2025. [Online]. Available: https://github.com/roboflow/rf-detr.

[65] O. Siméoni et al., *Dinov3*, 2025. arXiv: 2508.10104 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2508.10104.

[66] Google DeepMind, *Gemini Pro image generation model*, https://deepmind.google/models/gemini-image/pro/, Accessed: 2025-12-13, 2025.

[67] Y. Tian, Q. Ye, and D. Doermann, *Yolov12: Attention-centric real-time object detectors*, 2025. arXiv: 2502.12524 [cs.CV]. [Online]. Available: https://arxiv.org/abs/2502.12524.

A

Appendix 1

Table A.1: Hardware and OS Specifications

Component	**Specification**
Operating System	Ubuntu 24.04.3 LTS
CPU	AMD Ryzen 9 7900X (12-Core Processor)
GPU	NVIDIA GeForce RTX 4070 Ti SUPER
Memory (RAM)	64 GB DDR5 (4800 MT/s)

Table A.2: Per-scene detection performance (AP_{weighted}, i.e., weighted $AP_{50:95}$) for selected scenes across all model configurations.

Model	Model Adaptation	Labeling Strategy	Scene 019	Scene 058	Scene 090	Scene 125	Scene 156
RF (Medium)	Base	None	0.1471	0.5877	0.5974	0.5865	0.3440
	Standard Fine-tuning	ST	0.1488	0.6384	0.6433	0.5937	0.3797
		Ensemble	0.1201	0.6285	0.6214	0.6241	0.3761
		SSL-B	0.1138	0.6077	0.5840	0.5729	0.3469
		SAM3	0.1922	0.6161	0.5910	0.5744	0.4004
	Background-Context Fusion	ST	0.1633	0.6742	0.6043	0.5885	0.4413
		Ensemble	0.1329	0.6752	0.5977	0.6167	0.4436
		SSL-B	0.1251	0.6405	0.5312	0.5805	0.4180
		SAM3	0.2106	0.6647	0.5423	0.5566	0.4793
RF (Nano)	Base	None	0.1092	0.5602	0.5203	0.5645	0.3076
	Standard Fine-tuning	ST	0.1367	0.6031	0.5657	0.5680	0.3440
		Ensemble	0.1198	0.6219	0.6076	0.6035	0.3605
		SSL-B	0.0935	0.5425	0.4663	0.4809	0.2715
		SAM3	0.1783	0.6090	0.5557	0.5539	0.3856
	Background-Context Fusion	ST	0.1548	0.6438	0.5311	0.5653	0.4215
		Ensemble	0.1347	0.6649	0.5841	0.6039	0.4372
		SSL-B	0.1076	0.5680	0.4227	0.4737	0.3143
		SAM3	0.2023	0.6616	0.5283	0.5386	0.4602
Yolo11 (Large)	Base	None	0.1150	0.5202	0.4664	0.5363	0.2760
	Standard Fine-tuning	ST	0.1355	0.5659	0.5538	0.6109	0.3334
		Ensemble	0.0699	0.6045	0.5437	0.6125	0.3455
		SSL-B	0.0000	0.4743	0.3157	0.4411	0.2235
		SAM3	0.1930	0.6201	0.5008	0.5987	0.4169
	Background-Context Fusion	ST	0.1373	0.6026	0.5262	0.5734	0.3437
		Ensemble	0.0596	0.6143	0.4669	0.5680	0.3196
		SSL-B	0.0000	0.4977	0.3138	0.4512	0.2238
		SAM3	0.2105	0.6256	0.4875	0.5832	0.4350
Yolo11 (Nano)	Base	None	0.0265	0.2279	0.2549	0.2000	0.1433
	Standard Fine-tuning	ST	0.0475	0.4193	0.3378	0.4681	0.1986
		Ensemble	0.0360	0.4740	0.4143	0.5289	0.2556
		SSL-B	0.0000	0.1326	0.2485	0.1042	0.1168
		SAM3	0.0829	0.4814	0.4144	0.5072	0.2870
	Background-Context Fusion	ST	0.0621	0.4676	0.3235	0.4441	0.2393
		Ensemble	0.0459	0.5082	0.3945	0.4747	0.2914
		SSL-B	0.0000	0.2542	0.2457	0.1038	0.1441
		SAM3	0.0991	0.5123	0.3738	0.4452	0.3122

Table A.3: Summary statistics per model configuration (mean ± std) for core detection metrics over all scenes. AP refers to $AP_{50:95}$.

Model	Model Adaptation	Labeling Strategy	$\mathbf{AP}_{\mathbf{weighted}}$	$\mathbf{AP50}_{\mathbf{weighted}}$	AP	AP50
RF (Medium)	Base	None	0.4535 ± 0.1563	0.7028 ± 0.1602	0.4431 ± 0.1466	0.6736 ± 0.1598
	Standard Fine-tuning	ST	0.4779 ± 0.1533	0.7338 ± 0.1543	0.4592 ± 0.1438	0.6981 ± 0.1624
		Ensemble	0.4543 ± 0.1601	0.6648 ± 0.1758	0.4387 ± 0.1450	0.6403 ± 0.1680
		SSL-B	0.4402 ± 0.1590	0.6697 ± 0.1740	0.4247 ± 0.1479	0.6386 ± 0.1704
		SAM3	0.4763 ± 0.1441	0.7570 ± 0.1419	0.4485 ± 0.1349	0.7098 ± 0.1558
	Background-Context Fusion	ST	0.4912 ± 0.1475	0.7580 ± 0.1427	0.4732 ± 0.1397	0.7260 ± 0.1545
		Ensemble	0.4659 ± 0.1579	0.6811 ± 0.1720	0.4504 ± 0.1428	0.6583 ± 0.1661
		SSL-B	0.4485 ± 0.1553	0.6861 ± 0.1669	0.4343 ± 0.1445	0.6583 ± 0.1690
		SAM3	0.4883 ± 0.1382	0.7818 ± 0.1310	0.4639 ± 0.1310	0.7445 ± 0.1478
RF (Nano)	Base	None	0.4113 ± 0.1530	0.6797 ± 0.1719	0.4037 ± 0.1370	0.6502 ± 0.1647
	Standard Fine-tuning	ST	0.4370 ± 0.1492	0.6925 ± 0.1590	0.4181 ± 0.1463	0.6527 ± 0.1722
		Ensemble	0.4453 ± 0.1597	0.6547 ± 0.1777	0.4293 ± 0.1449	0.6295 ± 0.1693
		SSL-B	0.3719 ± 0.1592	0.5913 ± 0.1886	0.3591 ± 0.1458	0.5601 ± 0.1796
		SAM3	0.4655 ± 0.1443	0.7456 ± 0.1476	0.4387 ± 0.1351	0.7014 ± 0.1583
	Background-Context Fusion	ST	0.4474 ± 0.1442	0.7119 ± 0.1471	0.4277 ± 0.1433	0.6734 ± 0.1671
		Ensemble	0.4546 ± 0.1567	0.6686 ± 0.1727	0.4397 ± 0.1415	0.6472 ± 0.1666
		SSL-B	0.3742 ± 0.1562	0.5974 ± 0.1834	0.3624 ± 0.1428	0.5703 ± 0.1784
		SAM3	0.4758 ± 0.1379	0.7679 ± 0.1355	0.4518 ± 0.1304	0.7302 ± 0.1507
Yolo11 (Large)	Base	None	0.4227 ± 0.1656	0.6181 ± 0.1829	0.4059 ± 0.1576	0.5784 ± 0.1800
	Standard Fine-tuning	ST	0.4444 ± 0.1628	0.6245 ± 0.1822	0.4170 ± 0.1504	0.5850 ± 0.1811
		Ensemble	0.4148 ± 0.1685	0.5760 ± 0.2040	0.4008 ± 0.1530	0.5552 ± 0.1881
		SSL-B	0.3058 ± 0.1671	0.3954 ± 0.2015	0.2990 ± 0.1527	0.3852 ± 0.1868
		SAM3	0.4721 ± 0.1440	0.7307 ± 0.1449	0.4393 ± 0.1365	0.6810 ± 0.1560
	Background-Context Fusion	ST	0.4387 ± 0.1620	0.6121 ± 0.1840	0.4132 ± 0.1480	0.5742 ± 0.1779
		Ensemble	0.4064 ± 0.1663	0.5566 ± 0.2006	0.3922 ± 0.1476	0.5365 ± 0.1794
		SSL-B	0.3056 ± 0.1672	0.3937 ± 0.2032	0.2997 ± 0.1493	0.3863 ± 0.1839
		SAM3	0.4672 ± 0.1416	0.7261 ± 0.1464	0.4347 ± 0.1342	0.6743 ± 0.1573
Yolo11 (Nano)	Base	None	0.2406 ± 0.1367	0.4107 ± 0.1920	0.2339 ± 0.1313	0.3832 ± 0.1744
	Standard Fine-tuning	ST	0.2974 ± 0.1405	0.4827 ± 0.1878	0.2827 ± 0.1271	0.4482 ± 0.1758
		Ensemble	0.3446 ± 0.1523	0.5410 ± 0.1949	0.3252 ± 0.1334	0.5039 ± 0.1801
		SSL-B	0.1662 ± 0.1274	0.2456 ± 0.1795	0.1668 ± 0.1177	0.2428 ± 0.1631
		SAM3	0.3750 ± 0.1431	0.6395 ± 0.1695	0.3416 ± 0.1234	0.5748 ± 0.1646
	Background-Context Fusion	ST	0.3027 ± 0.1415	0.4861 ± 0.1867	0.2867 ± 0.1269	0.4515 ± 0.1737
		Ensemble	0.3423 ± 0.1501	0.5336 ± 0.1922	0.3213 ± 0.1309	0.4975 ± 0.1765
		SSL-B	0.1728 ± 0.1324	0.2532 ± 0.1858	0.1726 ± 0.1210	0.2492 ± 0.1659
		SAM3	0.3747 ± 0.1382	0.6390 ± 0.1604	0.3410 ± 0.1203	0.5753 ± 0.1579

Table A.4: Summary statistics per model configuration (mean ± std) for scale-specific metrics over all scenes. AP refers to $AP_{50:95}$.

Model	Model Adaptation	Labeling Strategy	$\mathbf{AP_{small}}$	$\mathbf{AP_{medium}}$	$\mathbf{AP_{large}}$
RF (Medium)	Base	None	0.1810 ± 0.1271	0.5260 ± 0.1460	0.6651 ± 0.5431
	Standard Fine-tuning	ST	0.2075 ± 0.1319	0.5500 ± 0.1387	0.6587 ± 0.5425
		Ensemble	0.1887 ± 0.1330	0.5393 ± 0.1595	0.6611 ± 0.5443
		SSL-B	0.1603 ± 0.1249	0.5191 ± 0.1513	0.6572 ± 0.5421
		SAM3	0.2140 ± 0.1199	0.5366 ± 0.1433	0.6192 ± 0.5342
	Background-Context Fusion	ST	0.2339 ± 0.1362	0.5588 ± 0.1343	0.6602 ± 0.5428
		Ensemble	0.2056 ± 0.1378	0.5515 ± 0.1573	0.6638 ± 0.5450
		SSL-B	0.1781 ± 0.1305	0.5276 ± 0.1475	0.6596 ± 0.5426
		SAM3	0.2429 ± 0.1245	0.5484 ± 0.1365	0.6150 ± 0.5343
RF (Nano)	Base	None	0.1549 ± 0.1111	0.4770 ± 0.1442	0.6268 ± 0.5279
	Standard Fine-tuning	ST	0.1839 ± 0.1269	0.4994 ± 0.1484	0.6267 ± 0.5350
		Ensemble	0.1778 ± 0.1301	0.5280 ± 0.1583	0.6541 ± 0.5420
		SSL-B	0.1125 ± 0.1066	0.4426 ± 0.1581	0.6223 ± 0.5331
		SAM3	0.2037 ± 0.1179	0.5260 ± 0.1434	0.6166 ± 0.5333
	Background-Context Fusion	ST	0.2024 ± 0.1314	0.5049 ± 0.1468	0.6298 ± 0.5354
		Ensemble	0.1935 ± 0.1339	0.5404 ± 0.1564	0.6583 ± 0.5435
		SSL-B	0.1238 ± 0.1120	0.4463 ± 0.1562	0.6226 ± 0.5353
		SAM3	0.2319 ± 0.1227	0.5378 ± 0.1384	0.6054 ± 0.5386
Yolo11 (Large)	Base	None	0.1376 ± 0.1181	0.4939 ± 0.1560	0.6402 ± 0.5358
	Standard Fine-tuning	ST	0.1748 ± 0.1303	0.5122 ± 0.1527	0.6154 ± 0.5419
		Ensemble	0.1485 ± 0.1300	0.4972 ± 0.1704	0.6399 ± 0.5433
		SSL-B	0.0620 ± 0.0916	0.3620 ± 0.1817	0.5724 ± 0.5424
		SAM3	0.2121 ± 0.1171	0.5372 ± 0.1399	0.6106 ± 0.5388
	Background-Context Fusion	ST	0.1787 ± 0.1275	0.5104 ± 0.1499	0.6142 ± 0.5378
		Ensemble	0.1446 ± 0.1216	0.4922 ± 0.1719	0.6263 ± 0.5405
		SSL-B	0.0618 ± 0.0889	0.3662 ± 0.1815	0.5620 ± 0.5412
		SAM3	0.2181 ± 0.1074	0.5343 ± 0.1396	0.5922 ± 0.5348
Yolo11 (Nano)	Base	None	0.0431 ± 0.0666	0.2628 ± 0.1405	0.4672 ± 0.5048
	Standard Fine-tuning	ST	0.0634 ± 0.0747	0.3475 ± 0.1350	0.4947 ± 0.4891
		Ensemble	0.0820 ± 0.0959	0.3998 ± 0.1530	0.5615 ± 0.5227
		SSL-B	0.0119 ± 0.0358	0.1703 ± 0.1369	0.3941 ± 0.4945
		SAM3	0.1211 ± 0.0906	0.4151 ± 0.1344	0.5253 ± 0.5184
	Background-Context Fusion	ST	0.0728 ± 0.0750	0.3576 ± 0.1358	0.4980 ± 0.4908
		Ensemble	0.0852 ± 0.0862	0.3990 ± 0.1487	0.5557 ± 0.5216
		SSL-B	0.0118 ± 0.0399	0.1801 ± 0.1435	0.4131 ± 0.4774
		SAM3	0.1212 ± 0.0828	0.4170 ± 0.1334	0.5391 ± 0.5186

Table A.5: Performance delta vs. base (per-scene calculation; mean ± std across scenes). Positive values indicate improvement over the corresponding base model of the same variant. ΔAP refers to $AP_{50:95}$.

Model	Model Adaptation	Labeling Strategy	Δ $AP_{weighted}$	Δ $AP50_{weighted}$	Δ AP	Δ AP50
RF (Medium)	Base	None	+0.0000 ± 0.0000	+0.0000 ± 0.0000	+0.0000 ± 0.0000	+0.0000 ± 0.0000
	Standard Fine-tuning	ST	+0.0244 ± 0.0263	+0.0311 ± 0.0439	+0.0161 ± 0.0612	+0.0245 ± 0.0777
		Ensemble	+0.0008 ± 0.0410	-0.0379 ± 0.0748	-0.0044 ± 0.0765	-0.0333 ± 0.1021
		SSL-B	-0.0133 ± 0.0239	-0.0331 ± 0.0449	-0.0184 ± 0.0599	-0.0350 ± 0.0766
		SAM3	+0.0228 ± 0.0408	+0.0543 ± 0.0528	+0.0054 ± 0.0727	+0.0361 ± 0.0884
	Background-Context Fusion	ST	+0.0377 ± 0.0365	+0.0553 ± 0.0533	+0.0300 ± 0.0693	+0.0524 ± 0.0893
		Ensemble	+0.0125 ± 0.0498	-0.0217 ± 0.0844	+0.0072 ± 0.0818	-0.0153 ± 0.1097
		SSL-B	-0.0050 ± 0.0337	-0.0166 ± 0.0542	-0.0089 ± 0.0663	-0.0153 ± 0.0879
		SAM3	+0.0348 ± 0.0477	+0.0790 ± 0.0629	+0.0207 ± 0.0787	+0.0709 ± 0.0965
RF (Nano)	Base	None	+0.0000 ± 0.0000	+0.0000 ± 0.0000	+0.0000 ± 0.0000	+0.0000 ± 0.0000
	Standard Fine-tuning	ST	+0.0258 ± 0.0433	+0.0128 ± 0.0542	+0.0144 ± 0.0673	+0.0025 ± 0.0851
		Ensemble	+0.0340 ± 0.0427	-0.0250 ± 0.0757	+0.0255 ± 0.0692	-0.0207 ± 0.0983
		SSL-B	-0.0394 ± 0.0370	-0.0884 ± 0.0569	-0.0446 ± 0.0626	-0.0901 ± 0.0802
		SAM3	+0.0542 ± 0.0383	+0.0658 ± 0.0555	+0.0350 ± 0.0653	+0.0512 ± 0.0843
	Background-Context Fusion	ST	+0.0361 ± 0.0552	+0.0322 ± 0.0755	+0.0240 ± 0.0729	+0.0232 ± 0.0983
		Ensemble	+0.0433 ± 0.0533	-0.0111 ± 0.0886	+0.0360 ± 0.0750	-0.0030 ± 0.1083
		SSL-B	-0.0370 ± 0.0451	-0.0823 ± 0.0677	-0.0413 ± 0.0655	-0.0799 ± 0.0884
		SAM3	+0.0646 ± 0.0477	+0.0882 ± 0.0694	+0.0481 ± 0.0714	+0.0800 ± 0.0928
Yolo11 (Large)	Base	None	+0.0000 ± 0.0000	+0.0000 ± 0.0000	+0.0000 ± 0.0000	+0.0000 ± 0.0000
	Standard Fine-tuning	ST	+0.0217 ± 0.0661	+0.0064 ± 0.0913	+0.0112 ± 0.0898	+0.0065 ± 0.1161
		Ensemble	-0.0078 ± 0.0756	-0.0421 ± 0.1285	-0.0051 ± 0.0959	-0.0232 ± 0.1347
		SSL-B	-0.1169 ± 0.0654	-0.2228 ± 0.1079	-0.1069 ± 0.0849	-0.1932 ± 0.1166
		SAM3	+0.0494 ± 0.0599	+0.1126 ± 0.0809	+0.0334 ± 0.0871	+0.1026 ± 0.1119
	Background-Context Fusion	ST	+0.0161 ± 0.0658	-0.0060 ± 0.0905	+0.0074 ± 0.0903	-0.0042 ± 0.1136
		Ensemble	-0.0163 ± 0.0778	-0.0615 ± 0.1236	-0.0137 ± 0.0983	-0.0419 ± 0.1320
		SSL-B	-0.1170 ± 0.0663	-0.2244 ± 0.1070	-0.1062 ± 0.0852	-0.1921 ± 0.1138
		SAM3	+0.0445 ± 0.0637	+0.1080 ± 0.0824	+0.0288 ± 0.0910	+0.0959 ± 0.1134
Yolo11 (Nano)	Base	None	+0.0000 ± 0.0000	+0.0000 ± 0.0000	+0.0000 ± 0.0000	+0.0000 ± 0.0000
	Standard Fine-tuning	ST	+0.0569 ± 0.0615	+0.0720 ± 0.0932	+0.0464 ± 0.0946	+0.0612 ± 0.1282
		Ensemble	+0.1041 ± 0.0825	+0.1303 ± 0.1238	+0.0913 ± 0.1081	+0.1207 ± 0.1505
		SSL-B	-0.0743 ± 0.0479	-0.1651 ± 0.0882	-0.0736 ± 0.0724	-0.1504 ± 0.0989
		SAM3	+0.1345 ± 0.0754	+0.2288 ± 0.0990	+0.1077 ± 0.0974	+0.1915 ± 0.1346
	Background-Context Fusion	ST	+0.0622 ± 0.0646	+0.0754 ± 0.0998	+0.0504 ± 0.0960	+0.0644 ± 0.1310
		Ensemble	+0.1017 ± 0.0824	+0.1229 ± 0.1251	+0.0874 ± 0.1068	+0.1142 ± 0.1502
		SSL-B	-0.0678 ± 0.0549	-0.1575 ± 0.0975	-0.0694 ± 0.0767	-0.1463 ± 0.1046
		SAM3	+0.1342 ± 0.0749	+0.2283 ± 0.1013	+0.1071 ± 0.0976	+0.1920 ± 0.1348

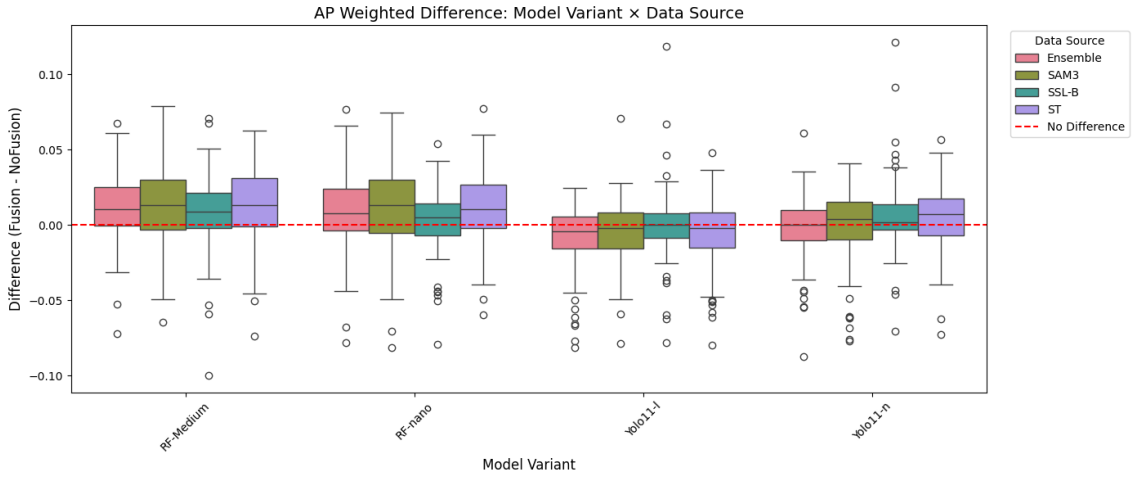


Figure A.1: Difference in AP_{weighted} between Background-Context Fusion (BF) and Standard Fine-tuning (SF), computed as $\Delta = AP^{\text{BF}}_{\text{weighted}} - AP^{\text{SF}}_{\text{weighted}}$ across scenes. Boxplots are grouped by model variant and labeling strategy (data source). The dashed red line indicates no difference ($\Delta = 0$).