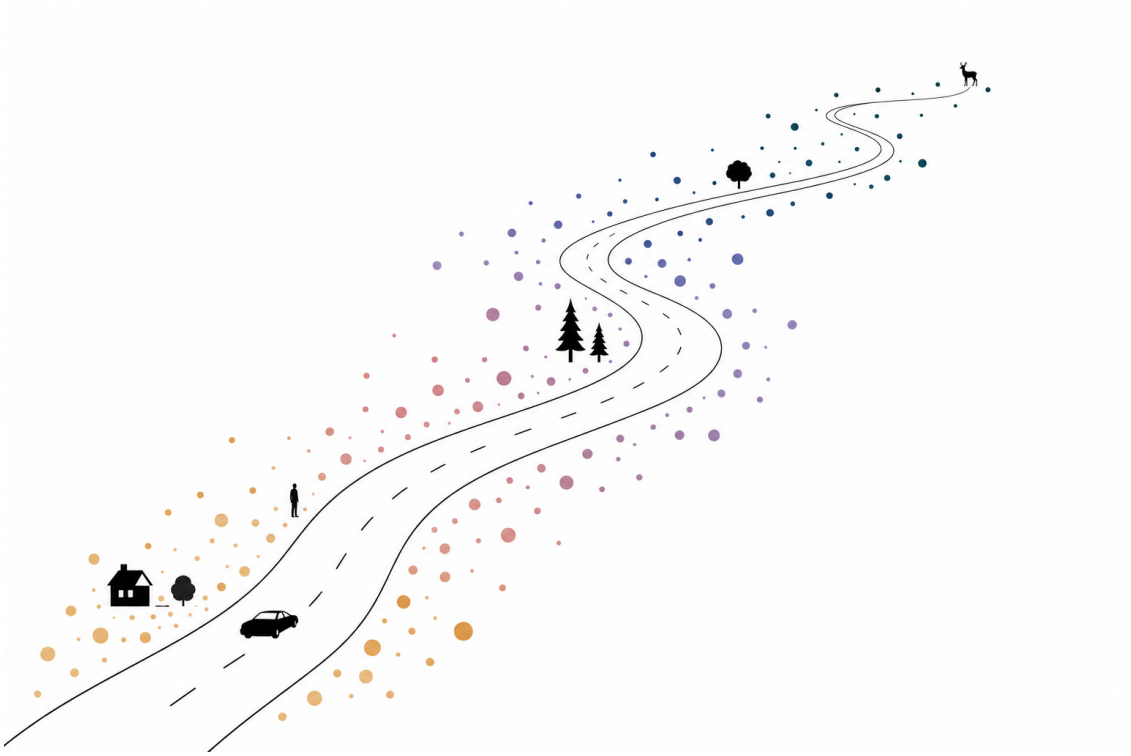

Novelty-Aware Frame Selection for Streaming Object Detection

A Centralized and Federated Study on Driving Data

Master's thesis in Complex Adaptive Systems

OSCAR LINDBERG

DEPARTMENT OF ELECTRICAL ENGINEERING

CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2026
www.chalmers.se

Master's thesis 2026

Novelty-Aware Frame Selection for Streaming Object Detection

A Centralized and Federated Study on Driving Data

OSCAR LINDBERG

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Electrical Engineering
Division of Antennas and Optical Networks
Chalmers University of Technology
Gothenburg, Sweden 2026

Novelty-Aware Frame Selection for Streaming Object Detection
A Centralized and Federated Study on Driving Data
OSCAR LINDBERG

Supervisor: Jonas Frankemölle, Scaleout Systems AB
Examiner: Alexandre Graell Amat, Department of Electrical Engineering

Master's Thesis 2026
Department of Electrical Engineering
Division of Antennas and Optical Networks
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: A winding road carries a vehicle from familiar urban daylight (warm tones, near) through weather, twilight, and night (cooling colors, distant) into rural unknown territory, illustrating the non-stationary stream of frames at the heart of this thesis.

Typeset in LaTeX
Printed by Chalmers Reproservice
Gothenburg, Sweden 2026

Novelty-Aware Frame Selection for Streaming Object Detection
A Centralized and Federated Study on Driving Data
OSCAR LINDBERG
Department of Electrical Engineering
Chalmers University of Technology

Abstract

Modern vehicle fleets produce continuous camera streams, but training on every frame is impractical under compute and bandwidth constraints. Many frames are redundant, while frames from altered lighting conditions, adverse weather, or unfamiliar road contexts carry most of the adaptation signal. This thesis studies *novelty-aware frame selection* for streaming object detection, where frames are scored by Mahalanobis distance to a periodically refreshed reference Gaussian. A bootstrap anchor keeps novelty tied to an initial urban daytime reference, and the scoring snapshot, reference, and threshold are refreshed together so scores stay meaningful as the detector evolves.

Using the Zenseact Open Dataset (ZOD) and an FCOS/ResNet-50 detector, the method is evaluated in centralized streaming and in a four-client federated setting. Each variant is paired with a random baseline at the same empirical acceptance rate to isolate selection quality from data volume. Results show three consistent patterns. The filter is *domain-sensitive*, with acceptance varying strongly across stream conditions. Periodic refresh is *essential for calibration*; without it, the static filter drifts from a 20% target acceptance rate to 77% and saturates on novel segments. At matched acceptance rate, detection gains are *positive but modest*. The primary practical benefit is systematic per-domain coverage, with frames from novel or under-represented driving conditions reliably prioritized over familiar content. The federated setting reproduces the same three patterns.

Keywords: streaming object detection, active learning, novelty-aware frame selection, Mahalanobis distance, domain shift, federated learning.

Contents

List of Figures **xi**

List of Tables **xiii**

1 Introduction **1**
1.1 Problem statement . . . 1
1.2 Research questions . . . 2
1.3 Project setting . . . 2
1.4 Contributions . . . 3
1.5 Thesis outline . . . 3

2 Background **5**
2.1 Object detection fundamentals . . . 5
2.1.1 Architecture . . . 5
2.1.2 Evaluation metrics . . . 6
2.2 Offline vs. streaming supervised learning . . . 6
2.3 Domain shift in road-scene data . . . 7
2.4 Selective training and novelty scoring . . . 7
2.4.1 From active learning to selective streaming . . . 7
2.4.2 Frame selection policies . . . 7
2.4.3 Feature-space novelty and Mahalanobis distance . . . 8
2.4.4 Bounded memory of accepted frames . . . 8
2.5 Representation drift and reference refresh . . . 9
2.5.1 Frozen reference . . . 9
2.5.2 Periodic refresh with a bootstrap anchor . . . 10
2.6 Buffer-based online optimization . . . 11
2.7 Federated learning . . . 12
2.7.1 Non-IID client data . . . 12
2.8 Related work . . . 13
2.8.1 Data selection for efficient training . . . 13
2.8.2 Continual and online streaming learning . . . 14
2.8.3 Out-of-distribution and novelty detection . . . 14
2.8.4 Federated learning under client heterogeneity . . . 14
2.9 Summary . . . 15

3 Methods **17**
3.1 Formal setting . . . 17

3.2 Dataset and domain structure . . . 18
3.2.1 Zenseact Open Dataset (ZOD) Frames . . . 18
3.2.2 Target classes . . . 18
3.2.3 Metadata . . . 18
3.3 Preprocessing . . . 19
3.4 Manifests and stream ordering . . . 19
3.4.1 Role of the manifest . . . 19
3.4.2 Bootstrap prefix . . . 20
3.4.3 Curated block ordering . . . 20
3.4.4 Temporal ordering . . . 21
3.5 Detector and embeddings . . . 21
3.5.1 Embeddings for novelty scoring . . . 22
3.6 Bootstrap phase . . . 23
3.6.1 Training . . . 23
3.6.2 Reference distribution and threshold calibration . . . 23
3.6.3 Initial scoring snapshot . . . 24
3.7 Streaming phase . . . 24
3.7.1 Online loop . . . 24
3.7.2 Training hyperparameters . . . 24
3.8 Filter policies . . . 24
3.8.1 Adaptive memory: window vs. reservoir . . . 26
3.8.2 Reference structure: single vs. two-reference . . . 27
3.8.3 Periodic refresh . . . 28
3.9 Federated streaming detection . . . 28
3.9.1 Client partitioning . . . 29
3.9.2 Federated schedule . . . 29
3.9.3 Aggregation . . . 30
3.9.4 Server-issued refresh . . . 30
3.10 Evaluation protocol . . . 31
3.10.1 Evaluation metrics . . . 31
3.10.2 Smoothed tail-K mAP . . . 31
3.10.3 Selection-behavior metrics . . . 31
3.10.4 Iso-accept comparisons . . . 32
3.10.5 Replication and statistical interpretation . . . 32
3.11 Experiment matrix . . . 32
3.11.1 Streaming experiment matrix . . . 32
3.11.2 Federated experiment matrix . . . 32

4 Results **35**
4.1 Centralized streaming . . . 35
4.1.1 Static filter baseline . . . 35
4.1.2 Routing behavior under periodic refresh . . . 35
4.1.3 Detection quality at iso-accept . . . 38
4.1.4 Memory size and bootstrap anchor . . . 40
4.1.5 Temporal replication . . . 42
4.1.6 Streaming summary . . . 42

4.2 Federated streaming 43
4.2.1 Per-client stream composition 44
4.2.2 Routing behavior under fleet-pooled refresh 45
4.2.3 Detection quality at iso-accept 48
4.2.4 Sensitivity ablations 50
4.2.5 Temporal and heavyLocal replication 52
4.2.6 Federated summary 53

5 Conclusion **55**
5.1 Findings 55
5.2 Limitations and future work 56

Bibliography **59**

A Supplementary results **I**
A.1 Streaming: per-block iso-accept decomposition I
A.2 Streaming: per-category mAP trajectories (window pair) I
A.3 Federated: per-category mAP trajectories (window pair) I

List of Figures

2.1 Mahalanobis threshold calibration in the feature embedding space . . 9
2.2 Sliding-window and reservoir memory structures 10
2.3 Representation drift: frozen reference vs. periodic refresh 11
2.4 One federated communication round under FedAvg 13

3.1 Representative frames for each stream block 22
3.2 Per-frame streaming pipeline (single-reference variant) 25
3.3 Per-frame streaming pipeline (two-reference ablation) 27

4.1 Static filter acceptance rate over the curated stream 36
4.2 Adaptive filter rolling acceptance rate on the curated stream 37
4.3 Per-block mean acceptance rate: adaptive variants and static filter (curated stream) 38
4.4 Iso-accept leaderboard: mAP vs. acceptance rate (curated stream) . . 39
4.5 Per-category mAP trajectories: reservoir two-reference (curated stream) 41
4.6 Adaptive filter rolling acceptance rate on the temporal manifest . . . 42
4.7 Stream composition of the curated client partition 44
4.8 Stream composition of the temporal client partition 45
4.9 Per-block mean acceptance rate: federated curated partition 46
4.10 Per-round per-client acceptance rates: curated partition 47
4.11 Per-round per-client acceptance rates: temporal partition 48
4.12 Iso-accept leaderboard: mAP vs. acceptance rate (federated, curated) 50
4.13 Per-category mAP trajectories: reservoir two-reference (federated, curated) 51

A.1 Per-category mAP trajectories: window two-reference (curated stream) II
A.2 Per-category mAP trajectories: window two-reference (federated, curated) III

List of Tables

3.1 Frame-level class prevalence across road-type categories 18
3.2 Curated stream ordering: thirteen post-bootstrap blocks 21
3.3 Bootstrap training hyperparameters 23
3.4 Streaming training hyperparameters 26
3.5 Domain-aligned client partition for the curated manifest 29
3.6 Federated training schedules 30
3.7 Streaming experiment matrix 33
3.8 Federated experiment matrix 34

4.1 Iso-accept comparison: headline adaptive variants (curated stream) . 40
4.2 Iso-accept comparison: ablation set ($M = R = 1\,000$, curated stream) 40
4.3 Per-client novelty ratio across partition-schedule combinations 49
4.4 Iso-accept comparison: headline adaptive variants (federated, curated) 49
4.5 Federated ablation: calibration percentile 52
4.6 Federated ablation: two-reference vs. single-reference 52

A.1 Per-block iso-accept Δ mAP summary (curated stream) I

1

Introduction

1.1 Problem statement

Camera-based perception is central to modern driver assistance and autonomous driving. *Object detection*—predicting class labels and bounding boxes for traffic participants—is both a standard benchmark task and a practical system component. Detectors are typically trained offline on a fixed dataset that is shuffled and revisited for many epochs. This protocol is convenient, but it does not reflect deployment settings in which data arrive continuously from a vehicle fleet. In such settings, observations are ordered and often *non-stationary* across road type, lighting, and geography, which is the characteristic regime of continual and streaming learning [1]. Training on every frame is also impractical because of compute, bandwidth, and storage constraints.

Consider a fleet of vehicles, each equipped with a detector trained on a fixed dataset during development. Once deployed, each vehicle encounters environments that may differ substantially from the training distribution—highways at night, rural roads in rain, unfamiliar urban layouts. Not every frame from these new environments is equally informative: many are redundant with what the model already knows, while others represent genuine domain shift. A practical system must decide, at the edge, which frames are worth retaining for model improvement and which can be safely discarded. This data selection problem calls for a novelty measure grounded in the model's current representational state, one that remains well-calibrated as the model continues to learn.

This thesis studies *novelty-aware frame selection* for streaming object detection. Incoming frames are scored by their Mahalanobis distance to a reference Gaussian fitted on detector feature embeddings [2], and those whose score exceeds a calibrated threshold are accepted for training. Novelty here refers to distributional deviation in feature space: the filter responds to domain shift in appearance, lighting, and road context. The bootstrap frames serve as a persistent anchor in the reference distribution, while the scoring snapshot and reference are periodically refreshed so that novelty remains well-defined as the detector adapts. The same mechanism is evaluated both centrally and in a federated setting with multiple clients that each observe a different slice of the stream.

1.2 Research questions

This thesis addresses five research questions.

1. Does the distribution-based filter produce *domain-sensitive* acceptance, with higher acceptance rates on stream segments that differ substantially from the bootstrap domain and lower rates on segments that resemble it?

2. How does *periodic refresh* of the scoring snapshot and the reference distribution affect the filter's behavior over a long non-stationary stream, compared with a static, bootstrap-only reference?

3. Among adaptive filter variants, how do the memory structure (sliding window vs. reservoir) and the reference structure (single Gaussian vs. two-reference min) shape acceptance behavior and detection quality, and what is the contribution of the bootstrap anchor?

4. At *matched empirical acceptance rate*, does novelty-guided selection improve overall and per-domain detection quality relative to random selection?

5. Does the same scoring mechanism transfer to a federated setting in which each client observes a distinct slice of the stream? In particular, does the filter produce calibrated, domain-sensitive acceptance rates per client across rounds? And does novelty-guided selection improve detection quality relative to random at matched acceptance rate in the federated setting?

1.3 Project setting

Experiments use the Zenseact Open Dataset (ZOD) *Frames* subset [3], which provides forward-facing RGB frames with 2D object annotations from real-world driving across Europe. Three foreground classes are retained: `Vehicle`, `Pedestrian`, and `VulnerableVehicle`. These cover the main traffic participants in the dataset and vary substantially in prevalence across driving contexts, providing a natural signal for domain shift.

Each stream begins with a fixed bootstrap prefix of 2 000 urban daytime frames. This prefix establishes the detector, the reference distribution, and the acceptance threshold. The remainder of the training split is then streamed once under one of two orderings: a *curated* ordering that arranges frames into thirteen contiguous blocks, progressing from urban daytime conditions through weather, illumination, and road-type shifts to a rural tail; and a *temporal* ordering that presents the same frames in capture-timestamp order.

Accepted frames are enqueued in a small training buffer and used for gradient updates. Adaptive filter variants periodically refresh the scoring snapshot, the reference, and the threshold together, while a static ablation freezes all three after bootstrap. The federated counterpart partitions the post-bootstrap stream across four

clients that collaborate via Federated Averaging [4]. The detector used throughout is FCOS [5] with a ResNet-50 FPN backbone.

1.4 Contributions

This thesis makes both methodological and empirical contributions.

1. A novelty-scoring protocol for streaming object detection using Mahalanobis distance to a reference built from bootstrap frames and a bounded memory of accepted frames. The scoring snapshot, reference, and acceptance threshold are recalibrated jointly at a fixed cadence, keeping novelty scores aligned with the evolving detector while the bootstrap anchor keeps novelty grounded in the initial training domain.

2. A systematic ablation of three design choices—memory structure (sliding window vs. reservoir), reference structure (single-reference Gaussian vs. two-reference min), and bootstrap anchoring—and their individual effect on acceptance behavior and detection quality. Each variant is evaluated against a static-filter ablation and volume-matched random baselines across two stream orderings and a federated four-client setting, replicated over three random seeds, with a supplementary sensitivity study on memory size ($M = R \in \{1\,000, 1\,500\}$).

3. An empirical analysis of how novelty-guided selection shapes acceptance behavior and detection quality under domain shift. Per-domain acceptance rates and score distributions are reported alongside overall and per-domain detection metrics (COCO mAP, per-class AP), for both the streaming and federated settings.

1.5 Thesis outline

Chapter 2 establishes the technical foundations—object detection, streaming and continual learning, domain shift, Mahalanobis-distance novelty scoring, and federated averaging—and situates the approach in the broader literature. Chapter 3 specifies the complete experimental design: the ZOD dataset and stream construction, the filter policies and their refresh procedure, the federated extension, and the evaluation protocol including the iso-accept comparison that isolates selection quality from data volume. Chapter 4 reports and interprets the empirical findings for both the centralized streaming and federated settings, including a discussion of the filter's limitations in this detection setting. Chapter 5 summarizes the main findings and contributions and discusses broader limitations and future directions.

2

Background

This chapter introduces the technical concepts underlying the experimental design in Chapter 3 and situates the approach in the broader literature.

2.1 Object detection fundamentals

Object detection is the task of localizing objects in an image and assigning a class label to each localized instance. For road-scene perception, the objects of interest typically include vehicles, pedestrians, cyclists, and traffic infrastructure.

A detector receives an image and returns a set of predictions

$$\hat{y} = \{(\hat{b}_i, \hat{c}_i, \hat{s}_i)\}_{i=1}^{N},$$

where $\hat{b}_i$ is a predicted bounding box, $\hat{c}_i$ is a class label, and $\hat{s}_i$ is a confidence score. During training, predictions are compared to ground-truth annotations to compute classification and localization losses.

2.1.1 Architecture

Modern detectors are typically organized as a feature extractor followed by multi-scale prediction. The feature extractor (*backbone*) computes hierarchical visual representations, and a feature pyramid network (FPN) [6] merges these across scales so that both small and large objects can be detected. Detection heads map each pyramid level to class and localization outputs. This decomposition is important in road scenes, where object scale varies widely with distance and camera perspective.

This thesis uses FCOS [5], a one-stage, anchor-free detector, with a ResNet-50 [7] backbone. Unlike two-stage detectors such as Faster R-CNN [8], which first generate region proposals and then classify them, FCOS produces class and localization predictions directly from dense feature maps without a separate region-proposal stage. Each spatial location on each FPN level is treated as a candidate object center. For each location, the model predicts class logits, four box offsets (l, t, r, b) relative to that location, and a centerness score that down-weights low-quality predictions near object boundaries. The final detection confidence combines class probability and centerness.

The anchor-free formulation eliminates manually designed anchor templates (the aspect-ratio and scale combinations required by anchor-based detectors), which simplifies training-time target assignment. Each location is assigned supervision based on whether it falls inside a ground-truth box and whether the box scale is compatible with that FPN level. At inference, dense predictions are aggregated across pyramid levels and filtered by non-maximum suppression.

2.1.2 Evaluation metrics

Detection quality is measured with intersection over union (IoU) and average precision (AP). IoU measures overlap between a predicted and a ground-truth box. AP summarizes detection quality for one class as the area under the precision–recall curve, where precision is the fraction of predicted detections that are correct and recall is the fraction of ground-truth objects that are detected. Mean AP (mAP) averages AP across classes.

This thesis reports the COCO-style mAP of Lin et al. [9]: mAP averaged over the ten IoU thresholds $\{0.50, 0.55, \ldots, 0.95\}$, denoted mAP@[0.50:0.95], together with the individual thresholds mAP@50 and mAP@75. Using multiple IoU thresholds separates coarse detection quality from strict localization quality. mAP@50 is more permissive and reflects class coverage, while mAP@75 is sensitive to box precision. This distinction is relevant when analyzing selective training, because a policy may improve class coverage without improving localization precision to the same degree.

2.2 Offline vs. streaming supervised learning

In conventional offline training, the dataset is a static pool; samples are shuffled and reused for many epochs. This yields strong final accuracy but assumes repeated access to all data and substantial compute budgets. In streaming training, data arrive as a sequence and the learner processes items in order, typically with strict limits on storage and revisits. The central question shifts from how to optimize over a fixed pool to how to update efficiently as new data arrive. The streaming regime overlaps conceptually with *continual* and *online learning*, where the learner must fit a non-stationary distribution under bounded memory and without the ability to revisit past data freely [1].

Single-pass processing is realistic for edge deployment. An embedded device in a vehicle has limited storage, limited compute for retraining, and may face privacy constraints that prevent retaining raw images. In such settings, each frame is processed once and is either used for a local update or discarded.

The protocol in this thesis uses two phases. In the *bootstrap phase*, the model undergoes multi-epoch training on a fixed initial prefix, establishing a capable detector and an initial reference distribution for novelty scoring. In the *online streaming phase*, the remaining stream is processed single-pass: each frame is scored, accepted or rejected, and, if accepted, used for a gradient update before being discarded. The bootstrap phase makes full use of an offline training budget before deployment, while

the online streaming phase operates within the single-pass and storage constraints of an in-service device.

2.3 Domain shift in road-scene data

Road-scene distributions vary substantially with driving context, including road type (city, highway, and rural roads differ in layout, object density, and speed), illumination (day, twilight, and night produce different contrast, color, and shadow patterns), and weather (rain, snow, and fog affect visibility, surface reflections, and sensor noise).

These shifts change both image appearance and class priors. In the ZOD dataset used in this thesis, for example, pedestrians appear in approximately 85% of urban frames but fewer than 2% of highway frames. A model initialized on urban daytime bootstrap data has seen little of night, rain, or rural conditions, and can only adapt to them if it receives relevant training signal as those contexts appear in the stream. Under limited compute and storage budgets, training uniformly allocates capacity to already-familiar content as much as to genuinely new conditions. Prioritizing frames from novel or under-represented contexts instead concentrates the training budget where it is most needed.

2.4 Selective training and novelty scoring

2.4.1 From active learning to selective streaming

Classical active learning assumes a pool of unlabeled samples and an annotation budget, and iteratively queries the most informative samples for labeling [10]. In this thesis, labels are already available. The constrained resources are training compute, memory, and communication bandwidth. The decision is therefore not *which sample to label*, but *which labeled sample to train on*. This framing is closer to coreset selection and to informativeness-based data pruning in deep learning [11, 12] than to traditional active learning. Like those methods, it operates on already-labeled data and asks which examples merit training. The novel constraint here is that selection must be made online under a single-pass budget, without retrospective access to the full pool.

2.4.2 Frame selection policies

A selection policy computes a score for each incoming frame and accepts or rejects it. The primary approach studied is *distribution-based filtering*: frames whose feature embedding is far from a reference distribution are accepted, treating distance as a proxy for novelty, while frames that are distributionally similar to the reference are rejected as redundant. The reference may be held fixed at bootstrap (*static filtering*) or updated periodically from a memory of accepted frames (*adaptive filtering*), with the trade-offs discussed in Section 2.5.

Random filtering, which accepts each frame independently with a fixed probability, serves as the matched-volume baseline. By setting its acceptance probability equal to the distribution filter's empirical rate, differences in detection outcomes can be attributed to selection quality rather than to data quantity.

2.4.3 Feature-space novelty and Mahalanobis distance

Let $\mathbf{x} \in \mathbb{R}^d$ be a feature embedding of a frame, and let $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ denote the mean and covariance of a reference embedding distribution. The Mahalanobis distance [13] is

$$d(\mathbf{x}) \;=\; \sqrt{(\mathbf{x}-\boldsymbol{\mu})^\top (\boldsymbol{\Sigma}+\epsilon\mathbf{I})^{-1}(\mathbf{x}-\boldsymbol{\mu})},$$

where $\epsilon > 0$ is a small regularization constant that ensures invertibility in high dimensions. A small $d(\mathbf{x})$ indicates that the frame is close to the reference and likely represents familiar content; a large $d(\mathbf{x})$ indicates that the frame is distant from the reference and more likely to represent novel or under-represented content.

The Mahalanobis distance has two properties that make it preferable to Euclidean distance for novelty detection in high-dimensional feature spaces. First, it accounts for the *correlation structure* of the reference distribution: a displacement along a high-variance direction, where the reference is already spread out, contributes less to the score than the same displacement along a low-variance direction. Second, it is *scale-invariant across feature dimensions*—because $\boldsymbol{\Sigma}^{-1}$ normalizes each dimension by its variance, no single feature axis dominates the distance merely because of its dynamic range.

Mahalanobis distance on a Gaussian fitted to penultimate-layer features has become a standard tool for out-of-distribution (OOD) detection in deep classifiers [2], often compared against simple softmax-confidence baselines [14]. The present thesis adapts this construction to streaming object detection: features come from a detector backbone rather than a classifier, the reference is fitted on the bootstrap set rather than on per-class training data, and the distance is used as a real-time acceptance score rather than as an offline OOD flag.

The acceptance threshold is calibrated from the reference frames themselves: the $(1-p)$-quantile of in-sample Mahalanobis distances serves as a data-driven cutoff that admits a target fraction p of reference frames, establishing a well-defined operating point without requiring an external validation set (Figure 2.1). Whether this threshold is held fixed or recalibrated as the reference is refreshed is explored in Chapter 3.

2.4.4 Bounded memory of accepted frames

An adaptive filter needs a bounded summary of the accepted stream to avoid storing every past embedding. Two standard choices are:

- A **sliding window** of the most recent M accepted frames. The oldest element is evicted when a new one is added. The window emphasizes recency.

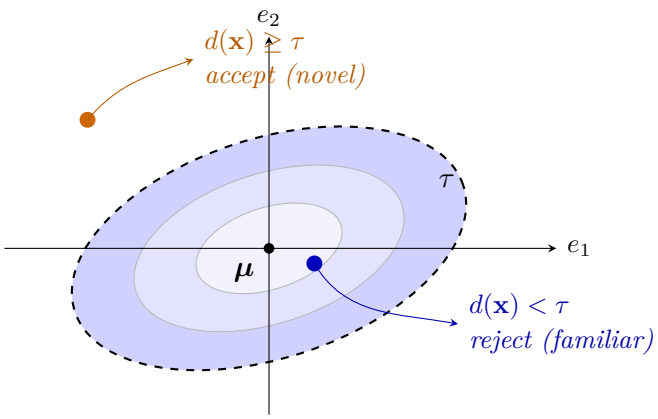

Figure 2.1: Mahalanobis threshold calibration in the feature embedding space (axes e_1, e_2 are the first two principal components of the 2048-dimensional backbone representation). The reference Gaussian fitted on the bootstrap frames has mean $\boldsymbol{\mu}$; concentric ellipses are iso-distance contours. The bold dashed contour is the threshold τ, set to the $(1-p)$-quantile of in-sample distances. Frames whose embedding falls *inside* τ (blue) are rejected as distributionally familiar; frames *outside* τ (orange) are accepted as novel.

- A **reservoir** of fixed size R maintained with Vitter's Algorithm R [15]. After n accepts, each past accept has equal probability $\min(1, R/n)$ of appearing in the reservoir, regardless of its position in the stream. The reservoir approximates a uniform sample of the accepted history.

The two choices encode different priors about what should be remembered: the sliding window biases toward recency, while the reservoir approximates a uniform sample of the entire accepted history (Figure 2.2). Their empirical consequences are compared in Chapter 4.

2.5 Representation drift and reference refresh

A distribution-based filter computes scores in a particular embedding space. If that embedding space changes during the stream, the meaning of a Mahalanobis distance changes with it. This is *representation drift*—a well-documented challenge when the model producing scores is itself being trained [16]. Under drift, a fixed threshold becomes difficult to interpret: the score scale can shift even when frame content does not, and two frames with the same Mahalanobis distance at different points in the stream may represent very different degrees of novelty (Figure 2.3). Two approaches to this problem are compared in this thesis.

2.5.1 Frozen reference

The simplest response is to *freeze* a snapshot of the model at the end of the bootstrap phase and compute every novelty score in that fixed embedding space. The live model continues to train on accepted frames, but the scoring snapshot is never

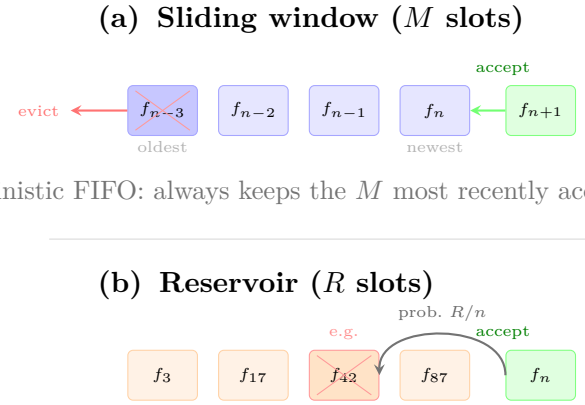

Figure 2.2: Bounded memory structures for the adaptive filter. *(a)* A sliding window of size M retains only the most recently accepted frames; each new acceptance evicts the oldest entry (FIFO). *(b)* A reservoir of size R maintains a uniform random sample of the entire accepted history; when the n-th frame is accepted it replaces a uniformly chosen slot with probability R/n, leaving the reservoir unchanged otherwise [15]. Non-sequential frame indices ($f_3, f_{17}, f_{42}, f_{87}$) illustrate that the reservoir can represent frames from any point in the stream. The window emphasizes recency; the reservoir provides global coverage.

updated, and the reference statistics $(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ are likewise held fixed. Scores are therefore exactly comparable across the entire stream, across stream orderings, and across federated clients. The filter is also stationary by design, which simplifies analysis.

Over a long non-stationary stream, however, the live detector diverges progressively from the frozen scoring model. The threshold was calibrated against the bootstrap's own score distribution, but the *live* model's embeddings of later frames are not bounded by that distribution. As the model adapts to newly accepted content, the gap widens, the empirical acceptance rate drifts upward, and the filter becomes progressively less selective. This is the static ablation in Chapter 3.

2.5.2 Periodic refresh with a bootstrap anchor

A more responsive alternative is to refresh the scoring snapshot and the reference periodically, jointly. At a fixed cadence, the current live model replaces the scoring snapshot, the reference frames are re-embedded under the new snapshot, the Gaussian $(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ is refit, and the threshold is recalibrated as a quantile of in-sample distances. Between refreshes the filter is again stationary, so scores are directly comparable within a refresh interval. This keeps novelty scores well-calibrated throughout the stream without the overhead of per-step re-estimation.

Periodic refresh introduces its own issue. If the reference is fit on *only* the accepted memory, it drifts toward whatever domain the policy has recently been accepting, and "novelty" becomes a moving target. Frames inside a homogeneous stream segment can become novel relative to a reference that has just been fit on the same

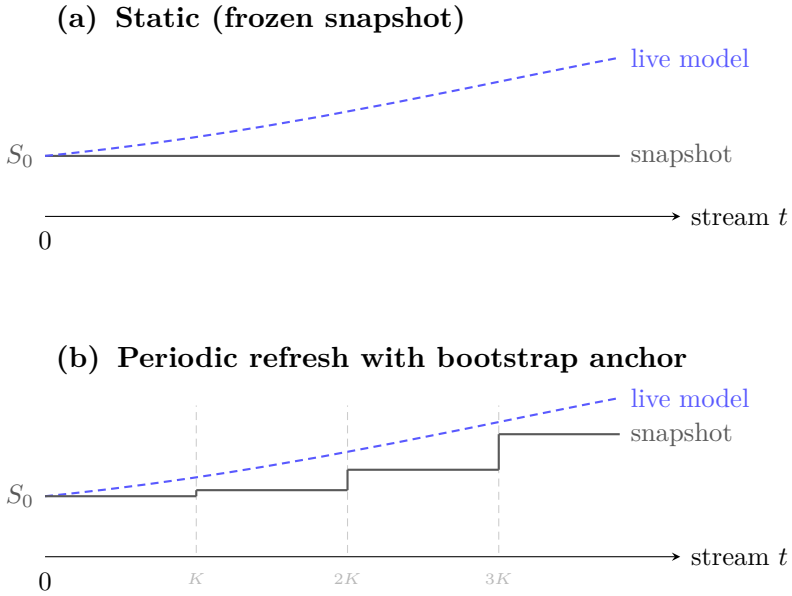

Figure 2.3: Representation drift: frozen reference vs. periodic refresh. In both panels the dashed blue curve is the live model evolving during streaming; the solid black line is the scoring snapshot used to compute novelty distances. *(a)* A frozen snapshot never updates, so the gap between the snapshot and the live model grows monotonically. Mahalanobis scores inflate and the empirical acceptance rate rises above the calibrated target p. *(b)* With periodic refresh, the snapshot is replaced by the current live model at each cadence mark (K, $2K$, $3K$, ...), keeping the gap small and scores well-calibrated within each interval. A bootstrap anchor (not depicted) ensures that the updated reference always includes the original bootstrap frames, preventing novelty from drifting to a moving target.

segment. To avoid this, the reference is *anchored* by including the bootstrap frames in the reference set at every refresh. Chapter 3 formalizes this scheme and includes an ablation in which the anchor is removed.

When the accepted memory drifts far from the bootstrap, the combined reference may no longer be well described by a single mode. Fitting one Gaussian to such a reference inflates its covariance and may place its mean in a low-density region, both of which reduce score discriminability. A two-reference variant, in which separate Gaussians are fit to the bootstrap and to the accepted memory and scored by the smaller Mahalanobis distance, is a natural response. This too is included as an ablation in Chapter 3.

2.6 Buffer-based online optimization

Deep detectors are trained with mini-batch gradient updates rather than per-sample online updates, since very small batches produce high-variance gradient estimates and interact poorly with batch normalization. In this thesis, accepted frames are collected in a bounded training buffer. When the buffer reaches capacity, the model

trains on its contents and the buffer is cleared. Compute cost is therefore roughly proportional to accepted items, a useful property when comparing filter policies under matched data budgets.

The buffer design limits how much the model can adapt. With a small buffer (for example, 32 frames) and single-pass streaming, each frame contributes only a handful of gradient steps before being discarded. There is no replay mechanism to revisit earlier data or to consolidate knowledge across domains. Aggregate detection metrics alone may therefore not fully reflect differences in selection quality. Properties of the selected set (domain coverage, score distributions, acceptance rates) provide complementary and more direct evidence of what each policy actually selects.

2.7 Federated learning

Federated learning trains a shared model across multiple clients without centralizing raw data [4, 17]. The typical loop is:

1. the server sends the current global model to participating clients;
2. each client performs local training updates on its own data;
3. the server aggregates client models into an updated global model.

The standard aggregation rule is Federated Averaging (FedAvg) [4]:

$$\theta^{(r+1)} = \sum_{k=1}^{K} w_k \, \theta_k^{(r+1)}, \qquad \sum_k w_k = 1,$$

where θ_k is client k's updated model and w_k is its aggregation weight. Weights are often proportional to the number of local training samples (Figure 2.4).

When combined with selective filtering, a natural choice is to weight each client by the number of *accepted* frames, since this reflects the client's actual training contribution. Clients in novel environments accept more frames, perform more gradient updates, and consequently have greater influence on the global model. This is a direct consequence of combining FedAvg with novelty-based selection.

2.7.1 Non-IID client data

In real deployments, client data are typically non-IID (non-independent and non-identically distributed): one vehicle may encounter mostly city traffic while another predominantly drives on highways. Heterogeneous client distributions can slow convergence and produce uneven per-domain performance [18], which has motivated aggregation variants such as FedProx [19] and SCAFFOLD [20] that correct for local drift. The experiments in this thesis use plain FedAvg to keep the aggregation mechanism fixed while the filter mechanism is varied.

The thesis uses a simulated federated setting in which each client receives a structured slice of the global stream: either a domain-aligned partition (curated manifest)

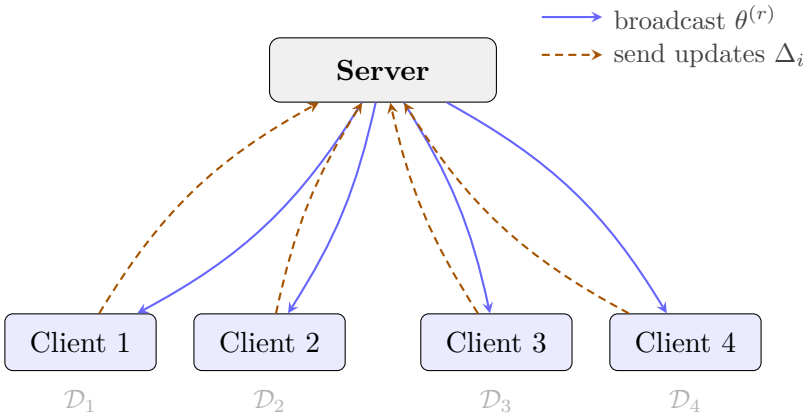

Figure 2.4: One federated communication round under FedAvg [4]. The server broadcasts the current global model $\theta^{(r)}$ to all participating clients (solid blue). Each client trains locally on its private dataset $\mathcal{D}_i$ and returns gradient updates Δ_i (dashed orange). The server aggregates these into $\theta^{(r+1)} = \sum_k w_k \, \theta_k^{(r+1)}$. In the federated variants of this thesis, the server additionally refreshes the shared filter state after each aggregation round and broadcasts it alongside $\theta^{(r+1)}$.

or a contiguous chronological quartile (temporal manifest). In both cases, different clients see different domain compositions, creating realistic imbalance while preserving full experimental control. In the adaptive federated variants, the scoring snapshot, the reference, and the threshold are refreshed on the *server* after each aggregation round. The server assembles a fleet-pooled reference by drawing an equal quota of accepted frame IDs from each client's local memory (window or reservoir), re-embeds them under the post-aggregation global model, and broadcasts the updated filter state to all clients. Every client therefore scores against the same reference within a round. Chapter 4 examines how this server-issued adaptation interacts with FedAvg aggregation and whether it produces meaningfully different per-client acceptance patterns across heterogeneous clients.

2.8 Related work

The present approach sits at the intersection of four bodies of literature: informativeness-based data selection, continual and streaming learning, out-of-distribution detection, and federated learning under client heterogeneity. This section positions the contribution within each and concludes by explaining where the present work stands relative to each.

2.8.1 Data selection for efficient training

A large body of work studies how to train on a subset of the available data without sacrificing accuracy. Active learning focuses on choosing which samples to label under a budget [10]; coreset methods select a weighted subset that approximates the gradients of the full set [11]; and gradient- or error-based scores such as GraNd and EL2N [12] rank labeled samples by their contribution to training loss. These

methods assume random access to a static dataset. The streaming setting considered here is more restrictive, as all selection decisions are made online in a single pass with no ability to revisit earlier frames. Unlike gradient-based scores, which require a backward pass, feature-space novelty scoring operates with a forward pass through a fixed snapshot.

2.8.2 Continual and online streaming learning

Continual learning studies learners that must fit a sequence of tasks or a non-stationary distribution while retaining prior knowledge [1]. Methods range from parameter regularization [21] to rehearsal from a small memory of past exemplars. The setup in this thesis shares the single-pass, non-stationary constraint but focuses on *input selection* rather than preventing forgetting: the model is not regularized to retain prior behavior, and the memory serves the scoring reference rather than replaying past gradients.

2.8.3 Out-of-distribution and novelty detection

The direct inspiration for the novelty score in this thesis is the Mahalanobis-distance OOD detector of Lee et al. [2], which fits class-conditional Gaussians to penultimate-layer features and scores inputs by their distance to the nearest class center. It is typically benchmarked against softmax-confidence baselines [14]. The present work adapts this construction to object-detection backbones and to a streaming selection loop, with two key differences: the reference is fitted on the full bootstrap set rather than per class, and the reference can be refreshed over time under the bootstrap-anchored adaptive scheme described in Section 2.5.

Using Mahalanobis scoring as a real-time frame filter introduces a challenge that does not arise in the original offline OOD setting: as the model trains on accepted frames, the embedding space drifts and a fixed reference becomes miscalibrated. The periodic refresh mechanism addresses this directly. Reis et al. [22] apply a related Mahalanobis score in continuous 32 Hz in-vehicle video streams, where the dominant challenge is redundancy among near-identical consecutive frames. In the present work, ZOD Frames provides individually curated keyframes rather than dense video, and the score serves as a domain-shift detector rather than a redundancy filter.

2.8.4 Federated learning under client heterogeneity

Federated learning over non-IID clients has been studied from several angles: convergence analysis [18], proximal regularization of local updates [19], variance-reduced aggregation [20], and broader surveys of the open problems [17]. The present thesis holds the aggregation rule fixed as plain FedAvg, varying only the per-client selection policy. Novelty-based selection nonetheless introduces implicit client weighting, as clients in novel domains accept more frames and consequently contribute more to the federated average without modifying the aggregation rule itself.

2.9 Summary

The methodology in this thesis integrates four elements: single-pass streaming training under domain shift, distribution-based frame selection with a periodically refreshed, bootstrap-anchored Mahalanobis reference, buffer-based online optimization, and evaluation under both centralized and federated settings. Relative to the literature surveyed above, it differs along four dimensions: it operates in a strictly single-pass setting without random access to a static pool; it treats distribution shift through input selection rather than through regularization or replay against forgetting; the Mahalanobis reference is fitted on a domain-specific bootstrap set and periodically refreshed rather than computed offline per class; and the implicit client weighting from novelty-based selection is an emergent property of combining FedAvg with per-client filtering, not a modification of the aggregation rule. Chapter 3 specifies the concrete implementation, data construction, filter policies, refresh procedure, and evaluation protocol.

3

Methods

This chapter specifies the experimental design for studying novelty-aware frame selection for streaming object detection: the data and stream construction, the detector, the training procedure, the filter policies and their refresh mechanism, the federated extension, and the evaluation protocol. The complete implementation, experiment configurations, and analysis notebooks are available at `https://github.com/scaleoutsystems/stream-active-FL`.

3.1 Formal setting

At time step t, the learner receives a labeled frame

$$z_t = (x_t,\, y_t),$$

where $x_t \in \mathbb{R}^{C\times H\times W}$ is an RGB image tensor and $y_t = \{(b_k, c_k)\}$ is the set of ground-truth bounding boxes $b_k \in \mathbb{R}^4$ (in x_1, y_1, x_2, y_2 format) with class labels c_k. A selection policy π observes z_t and the current model state θ_t and outputs

$$a_t = \pi(z_t, \theta_t) \in \{0, 1\},$$

with $a_t = 1$ (accept) or $a_t = 0$ (reject). The distribution-based policies defined below use only the embedding of x_t; labels y_t are not used by the selection policy. Accepted items are appended to a bounded training buffer $\mathcal{B}$ of capacity B; rejected items are discarded. When the buffer is full, the model is updated on buffered data and the buffer is cleared. After the stream ends, any remaining buffered items trigger a final update.

Training proceeds in two phases:

1. **Bootstrap.** Multi-epoch supervised training on a fixed prefix of N_{boot} frames, establishing the initial detector, the initial reference embedding distribution, and the initial acceptance threshold.

2. **Online streaming.** Single-pass processing of the remaining training frames under the selection policy. For the headline filter variants, the scoring model and reference distribution are refreshed periodically (Section 3.8.3), while the static ablation freezes them after bootstrap.

The bootstrap prefix and the stream segment are disjoint: no frame appears in both, so any improvement during streaming is attributable to genuinely new data rather than to revisits of bootstrap content.

3.2 Dataset and domain structure

3.2.1 Zenseact Open Dataset (ZOD) Frames

All experiments use the Zenseact Open Dataset (ZOD) [3], specifically the *Frames* subset: forward-facing RGB images with 2D object annotations from real-world driving collected across 14 European countries over two years. Each image is an individually curated keyframe rather than a consecutive sample from a continuous recording, so successive frames in the training split can originate from different drives and be separated by large gaps in capture time. The official train/validation split from the ZOD SDK is used, yielding approximately 90 000 training and 10 000 validation frames at the preprocessed resolution. Images are distributed in anonymized form (blurred faces and license plates).

3.2.2 Target classes

While ZOD defines ten foreground categories, the experiments use only three: Vehicle, Pedestrian, and VulnerableVehicle. All other categories are excluded from training and evaluation. Their prevalence across road types is shown in Table 3.1. Vehicle is present in virtually all frames regardless of road type, providing a stable detection target across conditions. Pedestrian and VulnerableVehicle prevalence varies strongly with road type, with Pedestrian frame prevalence dropping from 85% in urban scenes to under 2% on highways, making these two classes sensitive indicators of domain shift.

Table 3.1: Frame-level prevalence of the three target classes across road-type categories in the ZOD training split. A class is considered present if at least one annotated instance of that class appears in the frame.

Class	City	Art-urban	Highway	Art-rural	Sm-rural
Vehicle	99.0%	99.8%	99.8%	98.4%	96.9%
Pedestrian	85.1%	52.4%	1.8%	7.7%	22.8%
VulnerableVehicle	72.1%	52.0%	12.4%	11.6%	23.4%

3.2.3 Metadata

Each frame carries the following ZOD metadata fields used in this work:

- `road_type` $\in$ {city, arterial-urban, highway, arterial-rural, smaller-rural};
- `time_of_day` $\in$ {day, twilight, night};

- `scraped_weather` (sky / precipitation, sourced by ZOD from a weather service) ∈ {`clear-day`, `clear-night`, `partly-cloudy-day`, `partly-cloudy-night`, `cloudy`, `rain`, `snow`, `fog`, `wind`};
- `road_condition` (road surface annotation) ∈ {normal, wet, snow};
- capture timestamp.

For the purpose of grouping frames into stream blocks (Section 3.4.3), the two weather-related fields are folded into a coarse *weather bucket* (`clear`, `cloudy`, `fog`, `rain_wet`, or `snow`) by the priority rule

$$\text{snow} \succ \text{rain\_wet} \succ \text{fog} \succ \text{cloudy} \succ \text{clear},$$

where a frame matches the highest level that fires. (Because ZOD contains only ∼12 city-day-fog frames, the fog bucket is merged into cloudy within city-day, yielding a single `city_day_cloudy` block; see Section 3.4.3.)

The two highest buckets draw on both metadata fields: `snow` covers active-snowfall frames (`scraped_weather` contains `snow`) and post-snowfall frames with a snow-covered road surface (`road_condition` = snow); `rain_wet` covers frames with `scraped_weather` containing `rain` or `road_condition` = wet. The remaining buckets rely on `scraped_weather` alone. The `fog` bucket maps directly from that field; `cloudy` captures any value whose lowercase form contains `cloud` or `overcast`—in practice `cloudy`, `partly-cloudy-day`, and `partly-cloudy-night`. The `clear` bucket is the fallback, capturing `clear-day`, `clear-night`, and `wind`. None of these metadata fields are provided as inputs to the detector or to the selection policy. They serve only to construct manifest orderings and to slice analysis results by domain.

3.3 Preprocessing

Raw frames (3848×2168) are cropped to remove uninformative sky and ego-hood regions: the top 428 rows, bottom 588 rows, and 4 lateral pixels per side are removed, yielding a 3840×1152 crop. This is then resampled to 1600×480 (aspect ratio 10:3, Lanczos interpolation) and stored as JPEG. Annotations are read from the ZOD `object_detection` project; `unclear` instances are dropped. Bounding boxes are transformed into the cropped, resized coordinate system and stored in $x_1y_1x_2y_2$ format. A minimum-area filter of 64 px^2 is applied when loading data for training and evaluation, both to suppress label noise on tiny boxes and to keep the gradient signal concentrated on objects that the detector can plausibly localize.

3.4 Manifests and stream ordering

3.4.1 Role of the manifest

Each experiment points to a JSON manifest that lists frames with image paths, annotation paths, split membership, capture timestamps, and domain metadata.

The training stream order is exactly the order of training frames in the manifest, after removing the bootstrap prefix. This makes stream ordering explicit and reproducible: every variant on the same manifest sees the same frames in the same order, so differences in observed acceptance and per-domain behavior reflect the policy alone.

3.4.2 Bootstrap prefix

Each manifest follows the same structure: a bootstrap prefix of $N_{\text{boot}} = 2\,000$ training frames, followed by the remaining training frames in the chosen ordering. All 2 000 bootstrap frames have `road_type` = city and `time_of_day` = day, but cover the natural city-day weather mix without filtering: 725 `clear-day`, 832 `partly-cloudy-day`, 427 `cloudy`, and 16 `rain`/`wind` frames. The reference domain is therefore deliberately narrow (restricted to urban daytime), so that conditions outside it appear novel to the initial detector.

3.4.3 Curated block ordering

The default stream is the *city-day-curated* ordering: post-bootstrap training frames are grouped into thirteen contiguous blocks, progressing from the city-day bootstrap domain through a sequence of domain shifts (Table 3.2; Figure 3.1). Block labels are constructed from `road_type`, a coarse `time_of_day` category (`day` for day, `twi-night` for twilight or night), and the weather bucket of Section 3.2.3. As noted in Section 3.2.3, the `fog` bucket is merged into `cloudy` within city-day, yielding a `city_day_cloudy` block that encompasses both overcast and foggy city-day conditions. Within each block, frames are sorted by capture timestamp, so block-internal ordering corresponds to natural recording order. Every frame falls into exactly one block, and the bootstrap prefix plus the thirteen blocks partition the 89 972 ZOD training frames exactly ($2\,000 + 87\,972 = 89\,972$).

The ordering follows a progression of intentional shifts: weather variation within city-day (blocks 2–4), illumination shift to twilight and night (5–6), a within-urban road-type shift (7–8), a strong shift to highway (9–10), and a rural tail (11–13). Every block contains at least 1 000 frames, ensuring well-defined block-level statistics. Block sizes are nevertheless strongly imbalanced: the largest (`city_day_cloudy`, 21 624 frames) is roughly 21× the smallest (`city_day_snow`, 1 012 frames), reflecting the natural prevalence of each domain in the ZOD training split. A direct consequence is that the filter sees ∼24% of the entire stream in the opening block before the first weather shift occurs, and the curated ordering is therefore a deliberate compromise between domain balance (which would require subsampling) and faithfulness to the natural ZOD distribution. Note that block labels describe the metadata predicate, not the predominant visual appearance of every frame: `city_day_cloudy` includes partly-cloudy frames that may appear clear, and `city_day_rain_wet` includes frames with wet roads under an otherwise clear sky.

Table 3.2: City-day-curated stream ordering. Post-bootstrap training frames are grouped into thirteen contiguous blocks, progressing from the city-day bootstrap domain through a sequence of domain shifts. Frame counts exclude the 2 000 bootstrap frames. Block labels follow the convention ⟨road_type⟩_⟨time⟩_⟨weather bucket⟩ for the city-day weather sub-blocks, and ⟨road_type⟩_⟨time⟩ otherwise; `twi-night` aggregates `twilight` and `night`. Within each block, frames are sorted by capture timestamp.

#	Block label	Frames	%
1	`city_day_cloudy`	21 624	24.6%
2	`city_day_clear`	6 834	7.8%
3	`city_day_rain_wet`	7 079	8.0%
4	`city_day_snow`	1 012	1.1%
5	`city_twilight`	1 125	1.3%
6	`city_night`	5 337	6.1%
7	`arterial-urban_day`	14 807	16.8%
8	`arterial-urban_twi-night`	6 808	7.7%
9	`highway_day`	6 942	7.9%
10	`highway_twi-night`	2 862	3.3%
11	`arterial-rural_day`	5 873	6.7%
12	`arterial-rural_twi-night`	3 083	3.5%
13	`smaller-rural_all`	4 586	5.2%
	Total stream	**87 972**	100%

3.4.4 Temporal ordering

The city-day-temporal manifest reuses the same bootstrap prefix and the same post-bootstrap frame set as the curated manifest, but reorders the streaming portion strictly by capture timestamp. Domain transitions therefore occur at their natural sampling cadence rather than as engineered block boundaries. Pairing the curated and temporal manifests isolates the effect of stream ordering: the bootstrap, the validation set, and the population of stream frames are identical, so any differences between the two reflect the order in which the policy encounters domain shifts.

3.5 Detector and embeddings

The detector is FCOS [5] with a ResNet-50 [7] FPN [6] backbone, as implemented in `torchvision` (`fcos_resnet50_fpn`). The model is trained with four output classes (the three foreground classes plus background). The ResNet-50 backbone is initialized with ImageNet-pretrained weights [23], and the last three backbone stages are trainable while lower stages are frozen. Images are stored at 1600×480 (Section 3.3), matching the detector's expected input resolution, so no runtime rescaling is applied.

Figure 3.1: One representative frame per stream segment of the city-day-curated manifest, with target-class bounding boxes overlaid (Vehicle in blue, Pedestrian in red, VulnerableVehicle in orange; per-panel legend shown in the lower right). The top-left panel shows the bootstrap prefix; the remaining thirteen panels follow the stream order of Table 3.2 (left-to-right, top-to-bottom), with the panel title naming the block and the number of bounding boxes drawn. Within the early city-day rows, transitions hold structural metadata (`road_type`, `time_of_day`) constant while the weather bucket changes. Across the later rows, road type and time of day also shift, producing the intended sequence of domain changes.

3.5.1 Embeddings for novelty scoring

For distribution-based filtering, a 2048-dimensional embedding is extracted from the ResNet backbone and used as the frame representation for novelty scoring. A forward hook is registered on the ResNet backbone module to capture the spatial feature map produced by the final residual stage (2048 channels), and global average pooling reduces each spatial map to a single 2048-dimensional vector per frame. Embeddings are computed under `eval` mode with gradients disabled.

The scoring model used to compute embeddings is a *snapshot* of the detector, not the live model. After bootstrap, the snapshot equals the bootstrap-trained detector. In the headline adaptive variants, the snapshot is replaced periodically by the current live model (Section 3.8.3); in the static ablation, it remains the bootstrap snapshot for the entire stream. Snapshot embeddings are always computed on unaugmented frames, so the scorer sees deterministic inputs and identical frames produce identical scores.

3.6 Bootstrap phase

3.6.1 Training

The first $N_{\text{boot}} = 2\,000$ frames are used for multi-epoch supervised training with a shuffled data loader. Hyperparameters are listed in Table 3.3. Augmentation is applied only to this bootstrap phase. The embedding-collection and streaming phases use unaugmented frames to keep scoring deterministic. After bootstrap training, the detector is reasonably accurate on the bootstrap domain (urban daytime), which provides a stable starting point for streaming and a discriminative reference for novelty scoring.

Table 3.3: Bootstrap training hyperparameters.

Parameter	Value
Epochs	20
Batch size	8
Learning rate	4×10^{-4}
Optimizer	Adam (weight decay 10^{-4})
Augmentation	Horizontal flip ($p = 0.5$), color jitter
Mixed precision	Enabled

3.6.2 Reference distribution and threshold calibration

After bootstrap training, all N_{boot} bootstrap frames are passed through the bootstrap detector to obtain the reference mean $\boldsymbol{\mu}_0 \in \mathbb{R}^{2048}$ and covariance $\boldsymbol{\Sigma}_0 \in \mathbb{R}^{2048\times 2048}$. The novelty score for an embedding $\mathbf{x}$ is the Mahalanobis distance [13] to this reference, following the feature-space novelty construction used for out-of-distribution detection in deep classifiers [2]:

$$d(\mathbf{x};\, \boldsymbol{\mu}, \boldsymbol{\Sigma}) \;=\; \sqrt{(\mathbf{x}-\boldsymbol{\mu})^{\top}(\boldsymbol{\Sigma}+\epsilon\mathbf{I})^{-1}(\mathbf{x}-\boldsymbol{\mu})},$$

where $\epsilon = 10^{-5}$ is a small regularization constant that ensures invertibility of the full $2\,048 \times 2\,048$ sample covariance (which is rank-deficient when $N_{\text{boot}} < D$). The threshold τ is calibrated on the same regularized metric, keeping the percentile interpretation internally consistent. Applied to each bootstrap frame, this yields per-frame distances $d_i = d(\mathbf{x}_i;\, \boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0)$.

The acceptance threshold τ_0 is set so that a fraction p of the reference scores lies at or above the threshold:

$$\tau_0 \;=\; \text{Quantile}_{1-p}\Big(\{d_i\}\Big),$$

where $p \in (0, 1)$ is the calibration percentile, which equals the fraction of bootstrap frames whose score meets or exceeds τ_0, i.e., the fraction that would be accepted if the filter were applied back to its own reference set (values used per variant are listed in Table 3.7). For example, at $p = 0.20$, τ_0 equals the 80th percentile of bootstrap scores and exactly 20% of bootstrap frames lie at or above it.

3.6.3 Initial scoring snapshot

A deep copy of the bootstrap detector is taken, placed in `eval` mode with all gradients disabled, and used to compute embeddings for incoming stream frames. This snapshot is the initial scoring model. The static ablation keeps this snapshot fixed, while adaptive variants replace it periodically as part of the refresh procedure (Section 3.8.3).

3.7 Streaming phase

3.7.1 Online loop

After bootstrap, the remaining training frames are processed once in manifest order. For each frame:

1. The scoring snapshot computes a 2048-dimensional embedding.
2. The selection policy maps the embedding to an accept/reject decision.
3. Accepted frames are enqueued in the training buffer; rejected frames are discarded.
4. When the buffer is full ($|\mathcal{B}| = B$), the live model is trained on buffer contents and the buffer is cleared. This event is called a *buffer flush*.
5. For adaptive variants, every K buffer flushes a refresh is triggered (Section 3.8.3).

Figure 3.2 illustrates the complete per-frame flow.

3.7.2 Training hyperparameters

Streaming training hyperparameters are listed in Table 3.4. The streaming optimizer is reinitialized after bootstrap and does not inherit optimizer state from the bootstrap phase. Within each buffer, the loader is reshuffled at every local epoch, so two local epochs of a mini-batch loader produce eight gradient steps per flush at $B = 32$ and mini-batch size 8.

3.8 Filter policies

The experiments compare four families of selection policies. All variants share the bootstrap and streaming hyperparameters described above. They differ only in how acceptance decisions are made and, for distribution-based variants, in how the reference is maintained.

No filter. Every frame is accepted (subject to buffering). This is the upper bound on data availability and serves as the performance ceiling.

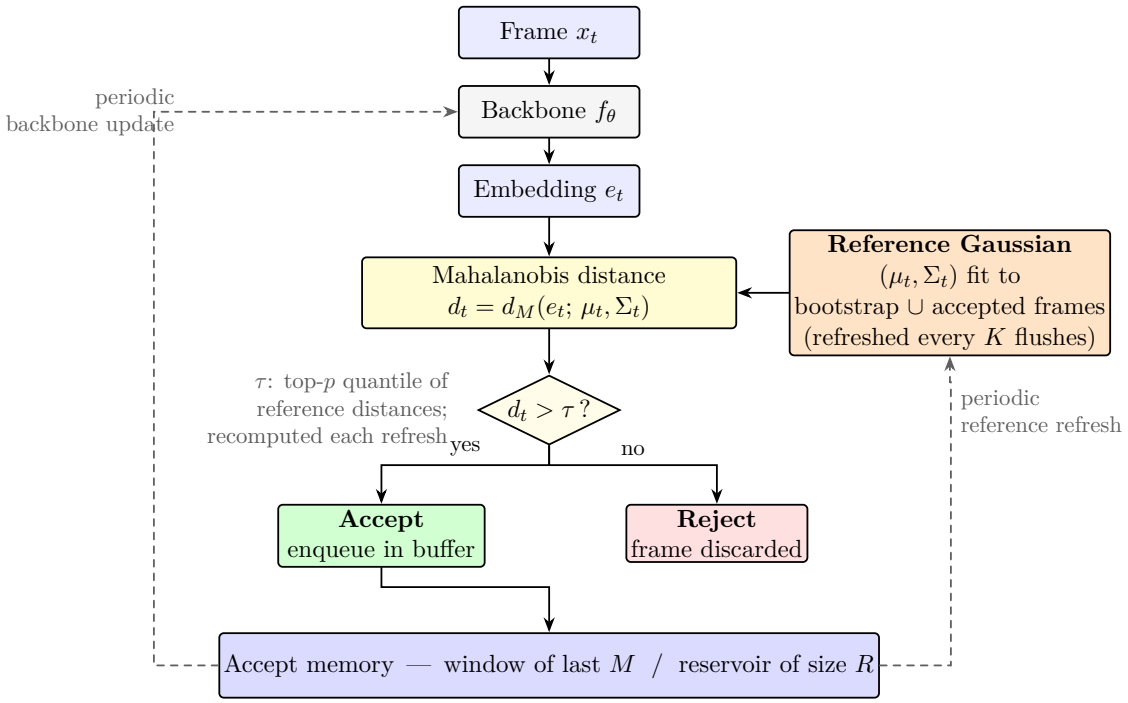

Figure 3.2: Per-frame streaming pipeline (single-reference variant). A stream frame x_t is encoded by the scoring snapshot f_θ to produce embedding e_t; the Mahalanobis distance to the current reference Gaussian (μ_t, Σ_t) is thresholded at τ to produce an accept or reject decision. Accepted frames enter the training buffer and accumulate in the accept memory (window or reservoir). Dashed arrows indicate periodic feedback: every K buffer flushes, the accept memory triggers a scoring-snapshot update and a refit of the reference Gaussian with recalibrated threshold τ (Section 3.8.3). The static ablation omits both feedback loops; the reference and snapshot are frozen after bootstrap.

Random filter. Each frame is accepted independently with probability p_{accept}. This baseline is domain-blind: it controls for data volume without using any information about novelty. Multiple acceptance probabilities are run in parallel so that each filter variant can be paired against a random run at matched empirical acceptance rate (Section 3.10.4).

Static distribution filter. The bootstrap-calibrated reference $(\boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0)$ and threshold τ_0 are held fixed for the entire stream. The embedding $\mathbf{x}_t$ of each stream frame is computed under the (frozen) bootstrap snapshot, and frame t is accepted iff $d(\mathbf{x}_t; \boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0) \geq \tau_0$, with the Mahalanobis distance defined in Section 3.6.2. This is the simplest distribution-based policy and is included as an ablation: it isolates the effect of *adaptation* when compared with the adaptive variants at the same percentile.

Adaptive distribution filter. The reference and the threshold are recomputed periodically from the bootstrap frames augmented with a memory of recently accepted frames. The scoring snapshot is replaced by the current live model at the

Table 3.4: Streaming training hyperparameters. These settings are shared across all filter variants.

Parameter	Value
Buffer capacity	$B = 32$ frames
Buffer training mode	Mini-batch
Mini-batch size	8
Local epochs per buffer	2
Streaming learning rate	1×10^{-4}
LR warmup	1 000 stream items (linear)
LR schedule	Cosine decay after warmup
Mixed precision	Enabled
Augmentation	Disabled (deterministic scoring)

same cadence. The adaptive family is parameterized along three axes (memory, scoring rule, bootstrap anchor) detailed in Sections 3.8.1–3.8.2, and the refresh procedure itself is described in Section 3.8.3. By default, the bootstrap frames are included in the reference set at every refresh, pinning the notion of novelty to the original known world. An ablation removes this anchor and fits the reference on the accepted memory alone (no-anchor variants in Table 3.7).

3.8.1 Adaptive memory: window vs. reservoir

The adaptive variants maintain an accepted-frame memory that is combined with the bootstrap to form the reference set at each refresh. Two memory schemes are compared:

- **Sliding window.** A FIFO deque of the most recent M accepted frames. When a new frame is accepted and the deque is full, the oldest frame is evicted. The memory therefore tracks recent stream content.

- **Reservoir.** A uniform random sample of size R drawn from all past accepted frames using Vitter's Algorithm R [15]. Each past accept has equal probability $\min(1, R/n)$ of being represented after n accepts, regardless of position in the stream.

The two schemes encode different priors about what an adaptive filter should remember: the window emphasizes recency and adapts quickly to local stream content, while the reservoir preserves global coverage and dilutes the influence of any single domain segment. The headline centralized variants use $M = R = 1\,500$; an additional ablation at $M = R = 1\,000$ probes sensitivity to memory size. Federated experiments use $M = R = 1\,500$ per client, with equal-quota pooling drawing 375 frames per client into the fleet-pooled reference at each server-side refresh, ensuring each client's contribution remains representative of its accepted history.

3.8.2 Reference structure: single vs. two-reference

When the accepted memory drifts away from the bootstrap, the union {bootstrap}∪{accepted} may no longer be well approximated by a single Gaussian. Fitting one Gaussian to a heterogeneous reference produces an inflated covariance and a mean that may fall in a low-density region of the data, which can reduce score discriminability. Two reference structures are therefore compared:

- **Single-reference (default).** One Gaussian $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ is fitted to the full reference set {bootstrap}∪{memory}. The score is the Mahalanobis distance to this Gaussian.

- **Two-reference (ablation).** Two Gaussians are fitted independently: $\mathcal{N}(\boldsymbol{\mu}_{\text{boot}}, \boldsymbol{\Sigma}_{\text{boot}})$ on the bootstrap (re-embedded under the current scoring snapshot) and $\mathcal{N}(\boldsymbol{\mu}_{\text{acc}}, \boldsymbol{\Sigma}_{\text{acc}})$ on the accepted memory. The score is

$$\min\Big(d(\mathbf{x}_t;\, \boldsymbol{\mu}_{\text{boot}}, \boldsymbol{\Sigma}_{\text{boot}}),\, d(\mathbf{x}_t;\, \boldsymbol{\mu}_{\text{acc}}, \boldsymbol{\Sigma}_{\text{acc}})\Big),$$

  so a frame is novel only when it is far from *both* known modes (Figure 3.3).

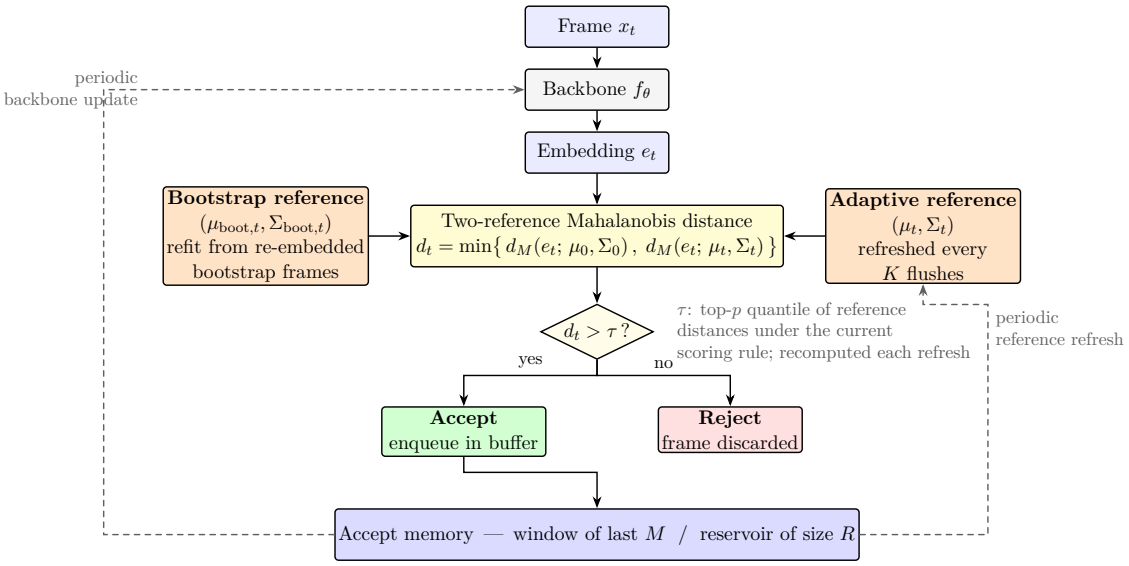

Figure 3.3: Per-frame streaming pipeline (two-reference ablation). Same per-frame flow as Figure 3.2, but two Gaussians are maintained independently: at each refresh, *both* the bootstrap frames and the accept memory are re-embedded under the new scoring snapshot, and each Gaussian is refit independently from the re-embedded frames. The bootstrap reference $(\mu_{\text{boot},t}, \Sigma_{\text{boot},t})$ therefore changes at every refresh (unlike a fixed bootstrap snapshot) because the scoring backbone changes. The novelty score is the minimum of the two Mahalanobis distances, so a frame is accepted only when it lies far from *both* known distributions. The threshold τ is recalibrated at each refresh as the $(1-p)$-quantile of per-frame min scores over the combined reference set (Section 3.8.3).

3.8.3 Periodic refresh

The headline adaptive procedure refreshes both the scoring snapshot and the reference statistics every K buffer flushes. Concretely, a refresh is triggered every K buffer flushes, at which point the following steps are performed:

1. **Snapshot the scorer.** The current live model is deep-copied, placed in `eval` mode with gradients disabled, and used as the new scoring snapshot. All subsequent embeddings (for stream frames and for re-embedding the reference set below) are computed under the new snapshot.

2. **Re-embed the reference set.** The bootstrap frames (if the bootstrap anchor is enabled) and the accepted memory are passed through the new snapshot to obtain fresh 2048-dimensional embeddings. Re-embedding ensures that all reference embeddings live in the same feature space as incoming stream frames; without it, the bootstrap statistics would drift relative to the new scorer.

3. **Refit the Gaussian(s).** In single-reference mode, one Gaussian is fitted to the union of re-embedded bootstrap and memory. In two-reference mode, two Gaussians are fitted independently as in Section 3.8.2. In the no-anchor ablation, the bootstrap is omitted and the Gaussian is fitted on the memory alone.

4. **Recalibrate the threshold.** Per-frame distances are recomputed for every frame in the (re-embedded) reference set under the active scoring rule (single-Mahalanobis or min-of-two-Mahalanobis), and the new threshold τ is set to the $(1 - p)$-quantile of those distances. Threshold recalibration at every refresh is essential: without it, a fixed τ inherited from bootstrap rapidly becomes incompatible with the rescaled reference distances after the embedding space has changed.

The default cadence is $K = 50$ buffer flushes, corresponding to one refresh per $K \cdot B = 1\,600$ accepted items. At a typical empirical acceptance rate of 20%, this is roughly every 8 000 stream frames. $K = 50$ was chosen as a practical balance between responsiveness and re-embedding cost, and sensitivity to this choice is not explored in this work. Shorter cadences keep the snapshot more current at the cost of more frequent re-embedding. Longer ones hold the reference fixed over larger stretches of stream at the cost of responsiveness.

3.9 Federated streaming detection

The federated extension simulates a fleet of vehicles that share the same bootstrap-trained detector and collaborate through Federated Averaging (FedAvg) [4]. The post-bootstrap stream is partitioned across four clients, each assigned a distinct data slice representing a different operational environment. All clients begin round 0 with the same scoring snapshot, the same reference $(\boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0)$, and the same threshold τ_0 from the bootstrap calibration of Section 3.6.2.

3.9.1 Client partitioning

For the curated manifest, the post-bootstrap stream is partitioned by *domain-aligned grouping*: the thirteen blocks of Section 3.4.3 are grouped into four clients in order of increasing domain distance from the bootstrap (Table 3.5).

Table 3.5: Domain-aligned client partition for the curated manifest. Frame counts exclude the 2 000 bootstrap frames. C0 is bootstrap-similar; C1 covers weather and illumination shifts within the city; C2–C3 extend to arterial-urban and then highway and rural environments.

Client	Blocks	Frames
C0	1–2 (city_day_cloudy, city_day_clear)	28 458
C1	3–6 (city_day rain/snow, city_twilight, city_night)	14 553
C2	7–8 (arterial-urban, day and twi-night)	21 615
C3	9–13 (highway, arterial-rural, smaller-rural)	23 346

C0 consists of the residual city-day frames (city-day-clear and city-day-cloudy) and matches the bootstrap domain most closely. C1 contains city frames under adverse weather and low-light conditions (rain, snow, twilight, night). C2 covers arterial-urban driving, and C3 groups highway and rural blocks. C1–C3 each represent a distinct dimension of novelty from the bootstrap: appearance shifts (C1) and road-type shifts of increasing scale (C2–C3).

For the temporal manifest, the post-bootstrap stream is split by capture timestamp into four equal-sized quartiles of 21 993 frames each, referred to as Q1–Q4 throughout Chapter 4. Unlike the curated partition, every client sees a mixture of road types and conditions. The main variation between quartiles reflects progression through the recording period. For example, Q4 carries roughly three times as many night frames as Q1, and Q3 has by far the most winter snow of any quartile.

3.9.2 Federated schedule

The experiments use two training schedules. Each schedule assigns every client a budget of 30 000 presented stream items, comparable in scale to each client's partition size (Table 3.6). Each client processes a contiguous slice of its partition in order, of which a fraction ρ is accepted for training.

All client partitions are smaller than the 30 000-item schedule budget (Table 3.5), so every client eventually exhausts its local data. The smallest partition (C1, 14 553 frames) stops contributing around round 15 of the default schedule, while the largest (C0, 28 458 frames) finishes near the end. After exhaustion, a client contributes no weight to aggregation in subsequent rounds, mirroring real deployments where vehicles operate at different data rates. The heavyLocal schedule trades aggregation frequency for local training volume (three times as many local items per round, three times fewer aggregation rounds), which isolates the effect of aggregation cadence at fixed total budget.

Table 3.6: Federated training schedules. Both schedules use the same four clients and partition strategy; the default schedule aggregates more frequently, while the heavyLocal schedule trains more data locally per round.

Parameter	Default	HeavyLocal
Number of clients	4	4
Number of rounds	30	10
Local items per round (presented)	1 000	3 000
Partition strategy	Domain-aligned (curated) / contiguous (temporal)	
Aggregation	FedAvg, weighted by accepted items	
Refresh cadence (adaptive)	Every round (server-issued)	

3.9.3 Aggregation

In each round, clients that accepted at least one frame contribute their updated live model, while clients with zero accepts contribute no weight. FedAvg aggregates client model parameters with weights proportional to the number of accepted frames in that round, and the aggregated model is distributed back to every client for the next round. This acceptance-weighted aggregation naturally prioritizes clients operating in novel environments: clients in unfamiliar domains accept more frames, perform more gradient updates, and consequently contribute more to the next global model.

3.9.4 Server-issued refresh

In the adaptive federated variants, the refresh of the scoring snapshot, the reference, and the threshold is performed on the *server* rather than per client, so that all clients share a consistent notion of novelty. Every K_{fed} rounds (set to 1 in all experiments reported here), after FedAvg aggregation, the server performs the refresh of Section 3.8.3 with one modification to how the accepted memory is assembled: rather than drawing from a single client's memory, the server builds a *fleet-pooled* reference by collecting an equal quota of accepted frame IDs from each client's local memory. Concretely, with a fleet pool size of 1 500 and four clients, each client contributes 375 frame IDs (1 500/4)—the most recent accepts for window-mode clients, or a random subsample of the reservoir for reservoir-mode clients. Each client accumulates its own window or reservoir of size 1 500 across rounds. This local memory is read at each refresh but is never reset or aggregated.

The fleet pool is then used in the standard refresh procedure: all bootstrap and fleet-pool frames are re-embedded under the post-aggregation global model (the new scoring snapshot), the Gaussian(s) are refit, and the threshold τ is recalibrated as the $(1-p)$-quantile of per-frame distances over the reference set. The resulting (snapshot, $\boldsymbol{\mu}, \boldsymbol{\Sigma}$, τ) is applied to every client policy, so all clients enter the next round scoring against the same reference.

This design preserves a *single* notion of novelty across the fleet. Per-client acceptance rates can differ because clients see different data, but they are all evaluated against the same reference, so accept-rate differences reflect each client's data composition rather than divergent per-client references. The equal-quota pool also prevents

dominant clients (e.g., the city client with more accepted frames) from diluting the reference with familiar content. The static federated ablation (no refresh) uses the bootstrap reference and threshold for all rounds and is otherwise identical.

3.10 Evaluation protocol

3.10.1 Evaluation metrics

The live model is evaluated on the full ZOD validation split (∼10 000 frames). Reported metrics follow the COCO protocol [9]: mAP averaged over IoU thresholds $\{0.50, 0.55, \ldots, 0.95\}$, mAP@50, mAP@75, and per-class AP for Vehicle, Pedestrian, and VulnerableVehicle. Evaluation is performed every 5 000 stream items in centralized runs and after every communication round in federated runs.

In addition to overall validation mAP, per-domain mAP is computed by restricting the validation set to frames whose metadata match a particular value of one of four domain dimensions:

- `stream_block`: the thirteen curated blocks of Section 3.4.3;
- `road_type`: city, arterial-urban, highway, arterial-rural, smaller-rural;
- `time_of_day`: day, twilight, night;
- `road_condition`: normal, wet, snow.

Per-domain mAP is recorded at every checkpoint, so the trajectory of detection quality within each domain can be tracked across the full stream.

3.10.2 Smoothed tail-K mAP

A single late checkpoint can be noisy because it depends on which frames fell into the last few buffers and on the cosine-decay learning rate at the end of the stream. The headline summary statistic is therefore the *smoothed tail-5 mAP*: the average of the last $K = 5$ checkpoint mAP values for a run (the final 25 k stream items in centralized runs, the final five communication rounds in federated runs). This smoothed value captures the model's late-stream performance level rather than any single checkpoint. Final-checkpoint mAP is also reported alongside, so that any difference between the smoothed and final values is visible.

3.10.3 Selection-behavior metrics

In addition to detection metrics, the following selection statistics are reported: overall acceptance rate, acceptance rate by stream block, road type, and time of day, score distributions per block and domain, cumulative accepted frames over the stream, and the number of optimizer steps executed (proportional to accepted frames).

3.10.4 Iso-accept comparisons

Because filter variants and random baselines often differ in the absolute number of accepts, headline detection comparisons are made at *matched empirical acceptance rate*: for each filter variant, the random baseline with the closest empirical acceptance rate is selected as its partner. We refer to this as an *iso-accept comparison*. The mAP difference between the two then measures *selection quality*, isolated from the data-volume effect. Pairings are listed in the *Partner* column of Table 3.7.

3.10.5 Replication and statistical interpretation

All experiments are replicated with three random seeds ($\{42, 43, 44\}$). Results are reported as seed means with standard deviations, and figures include cross-seed standard-deviation bands. Three seeds are sufficient to reveal instability but too few for reliable significance testing. All iso-accept gaps should therefore be interpreted in light of the seed-level variability reported in Tables 4.1 and 4.4. A gap smaller than the seed standard deviation of either variant in the pair is not reliable evidence of superiority. Section 4.1.3 identifies the cases where this applies.

3.11 Experiment matrix

3.11.1 Streaming experiment matrix

Table 3.7 lists all centralized streaming variants. The refresh cadence is $K = 50$ buffer flushes and streaming evaluation occurs every 5 000 items. Adaptive variants are run at two memory sizes: $M = R = 1\,500$ for the headline configuration and $M = R = 1\,000$ as a tighter-memory ablation set.

3.11.2 Federated experiment matrix

Table 3.8 lists all federated variants. The setup follows Section 3.9: four clients with domain-aligned (curated) or contiguous-quartile (temporal) partitions, per-client memory of 1 500, and server-side fleet-pooled refresh every round. Two schedules are used—*default* (30 rounds of 1 000 items per client) and *heavyLocal* (10 rounds of 3 000 items per client, matched total budget)—and each variant is replicated across three seeds.

The four headline adaptive filters are the single- and two-reference variants of both the window and reservoir families at $p = 0.20$. A static $p = 0.20$ filter and a no-filter ceiling complete the ablation set on the curated manifest. The temporal manifest is run with the reservoir two-reference filter only ($p \in \{0.15, 0.20\}$), since the mixed per-client domain composition of the temporal partition makes finer ablations redundant. An additional tighter percentile ($p = 0.10$) is included as a sensitivity ablation on the curated default condition.

Table 3.7: Streaming experiment matrix for the curated and temporal orderings. Each row corresponds to one variant replicated across three seeds. p denotes the calibration percentile (Section 3.6.2); *Mem.* is the adaptive memory structure and its size $M = R$; *Ref.* is the reference structure (single-reference Gaussian or two-reference min, Section 3.8.2); *Anchor* indicates whether the bootstrap is included in the reference set. *Partner* is the iso-accept random baseline paired against each filter variant for the matched-volume detection comparison (Section 3.10.4); abbreviated as `r{`q`}` where q is the random acceptance probability. The headline adaptive set ($M = R = 1\,500$) is listed first; the $M = R = 1\,000$ ablation set sweeps the two-reference and no-anchor variants at a tighter memory budget.

Variant	Manifest	Family	p	Mem. (M=R)	Ref.	Anchor	Partner
Reference baselines							
`no_filter`	curated	none	–	–	–	–	–
`random_p{17,21,23,26,27,33,77}`	curated	random	–	–	–	–	–
Static distribution filter							
`static_p20`	curated	dist (static)	0.20	none	single	yes	`r77`
Adaptive window — headline ($M = 1\,500$)							
`window_p20_m1500`	curated	dist (window)	0.20	window, 1 500	single	yes	`r26`
`window_p20_twoRef_m1500`	curated	dist (window)	0.20	window, 1 500	two	yes	`r27`
Adaptive reservoir — headline ($R = 1\,500$)							
`reservoir_p20_m1500`	curated	dist (reservoir)	0.20	reservoir, 1 500	single	yes	`r23`
`reservoir_p20_twoRef_m1500`	curated	dist (reservoir)	0.20	reservoir, 1 500	two	yes	`r21`
Adaptive window — ablation set ($M = 1\,000$)							
`window_p20`	curated	dist (window)	0.20	window, 1 000	single	yes	`r33`
`window_p20_twoRef`	curated	dist (window)	0.20	window, 1 000	two	yes	`r33`
`window_p20_noBoot`	curated	dist (window)	0.20	window, 1 000	single	no	`r33`
Adaptive reservoir — ablation set ($R = 1\,000$)							
`reservoir_p20`	curated	dist (reservoir)	0.20	reservoir, 1 000	single	yes	`r21`
`reservoir_p20_twoRef`	curated	dist (reservoir)	0.20	reservoir, 1 000	two	yes	`r21`
`reservoir_p20_noBoot`	curated	dist (reservoir)	0.20	reservoir, 1 000	single	no	`r17`
Cross-stream-order replication (temporal)							
`no_filter`	temporal	none	–	–	–	–	–
`random_p{21,28,31}`	temporal	random	–	–	–	–	–
`window_p20`	temporal	dist (window)	0.20	window, 1 000	single	yes	`r28`
`window_p20_twoRef`	temporal	dist (window)	0.20	window, 1 000	two	yes	`r31`
`reservoir_p20`	temporal	dist (reservoir)	0.20	reservoir, 1 000	single	yes	`r21`
`reservoir_p20_twoRef`	temporal	dist (reservoir)	0.20	reservoir, 1 000	two	yes	`r21`

Table 3.8: Federated experiment matrix. The *default* schedule is 30 rounds of 1 000 items per client; the *heavyLocal* schedule is 10 rounds of 3 000 items per client. Adaptive variants use per-client memory of size 1 500 (window or reservoir) and a fleet pool of 1 500 at server-side refresh. The four headline adaptive filters (single-/two-reference, window/reservoir at $p = 0.20$) constitute the main comparison set; all other rows are sensitivity ablations. *Partner* is the iso-accept random baseline; abbreviated as `r{`q`}` where q is the random acceptance probability (manifest and schedule are inherited from the row; `-` indicates no matched-volume comparison for that variant). Column conventions otherwise mirror Table 3.7.

Variant	Manifest	Schedule	Family	p	Mem.	Ref.	Partner
Curated manifest, default schedule							
`fed_no_filter`	curated	default	none	–	–	–	–
`fed_random_p{7,12,15,18,77}`	curated	default	random	–	–	–	–
`fed_static_p20`	curated	default	dist (static)	0.20	none	single	`r77`
`fed_adaptive_window_p20`	curated	default	dist (window)	0.20	window	single	`r12`
`fed_adaptive_window_p20_twoRef`	curated	default	dist (window)	0.20	window	two	`r12`
`fed_adaptive_window_p10`	curated	default	dist (window)	0.10	window	single	–
`fed_adaptive_reservoir_p20`	curated	default	dist (reservoir)	0.20	reservoir	single	`r18`
`fed_adaptive_reservoir_p20_twoRef`	curated	default	dist (reservoir)	0.20	reservoir	two	`r15`
`fed_adaptive_reservoir_p10`	curated	default	dist (reservoir)	0.10	reservoir	single	–
`fed_adaptive_reservoir_p10_twoRef`	curated	default	dist (reservoir)	0.10	reservoir	two	–
Curated manifest, heavyLocal schedule							
`fed_no_filter_heavyLocal`	curated	heavyLocal	none	–	–	–	–
`fed_random_p{16,21,26}_heavyLocal`	curated	heavyLocal	random	–	–	–	–
`fed_adaptive_window_p20_heavyLocal`	curated	heavyLocal	dist (window)	0.20	window	single	`r26`
`fed_adaptive_window_p20_twoRef_heavyLocal`	curated	heavyLocal	dist (window)	0.20	window	two	`r26`
`fed_adaptive_reservoir_p20_heavyLocal`	curated	heavyLocal	dist (reservoir)	0.20	reservoir	single	`r26`
`fed_adaptive_reservoir_p20_twoRef_heavyLocal`	curated	heavyLocal	dist (reservoir)	0.20	reservoir	two	`r26`
`fed_adaptive_reservoir_p15_heavyLocal`	curated	heavyLocal	dist (reservoir)	0.15	reservoir	single	–
`fed_adaptive_reservoir_p15_twoRef_heavyLocal`	curated	heavyLocal	dist (reservoir)	0.15	reservoir	two	–
`fed_adaptive_reservoir_p10_twoRef_heavyLocal`	curated	heavyLocal	dist (reservoir)	0.10	reservoir	two	–
Temporal manifest, default schedule							
`fed_no_filter_temporal`	temporal	default	none	–	–	–	–
`fed_random_p{15,19}_temporal`	temporal	default	random	–	–	–	–
`fed_adaptive_reservoir_p15_twoRef_temporal`	temporal	default	dist (reservoir)	0.15	reservoir	two	`r15`
`fed_adaptive_reservoir_p20_twoRef_temporal`	temporal	default	dist (reservoir)	0.20	reservoir	two	`r19`
Temporal manifest, heavyLocal schedule							
`fed_no_filter_temporal_heavyLocal`	temporal	heavyLocal	none	–	–	–	–
`fed_random_p{25,30}_temporal_heavyLocal`	temporal	heavyLocal	random	–	–	–	–
`fed_adaptive_reservoir_p15_twoRef_temporal_heavyLocal`	temporal	heavyLocal	dist (reservoir)	0.15	reservoir	two	–
`fed_adaptive_reservoir_p20_twoRef_temporal_heavyLocal`	temporal	heavyLocal	dist (reservoir)	0.20	reservoir	two	`r30`

4

Results

This chapter reports the empirical results of the experiments defined in Chapter 3. Section 4.1 covers the centralized streaming experiments: routing behavior across domains, acceptance dynamics under periodic refresh, and detection quality at matched acceptance rate. Section 4.2 then presents the federated results.

4.1 Centralized streaming

4.1.1 Static filter baseline

The static filter freezes the scoring snapshot, the reference, and the threshold after bootstrap (Section 3.8), so its per-window acceptance rate directly reflects the empirical novelty of each stream segment relative to the bootstrap distribution. This makes it the cleanest variant to inspect first: with no moving reference, every acceptance spike is directly interpretable as domain distance from bootstrap.

Figure 4.1 shows the per-window acceptance rate alongside the rolling stream composition. The acceptance rate is already above the calibrated 20% target in the opening city blocks and rises rapidly, exceeding 0.8 before block 1 ends. The sharpest peaks—approaching 1.0—occur at the twilight and night segments (`city_twilight`, `city_night`, and every twi-night block downstream), where the illumination shift moves nearly all embeddings far from the fixed bootstrap Gaussian. Later daytime blocks show rates of 0.7–0.9, well above the calibrated target. The mean acceptance rate across the stream is $\rho \approx 0.77$, nearly four times the calibrated $p = 0.20$ target. Since τ_0 was calibrated on bootstrap embeddings, stream segments whose embeddings lie far from the bootstrap Gaussian systematically exceed the threshold, and the acceptance rate grows as successive novel segments arrive. This drift is what periodic refresh is designed to correct (Section 2.5).

4.1.2 Routing behavior under periodic refresh

The adaptive variants refresh the scoring snapshot, the reference, and the threshold every $K = 50$ buffer flushes (Section 3.8.3), so their acceptance rate reflects novelty relative to a rolling mixture of bootstrap and recently accepted frames, rather than the fixed bootstrap alone.

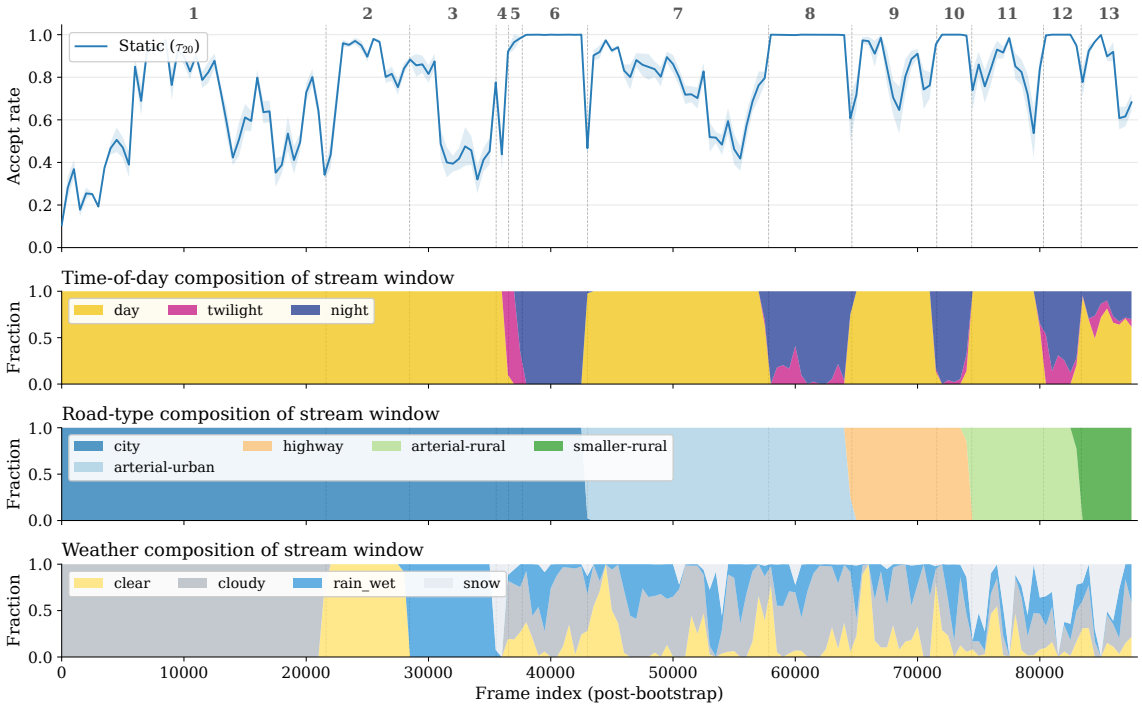

Figure 4.1: Static filter acceptance rate over the curated stream (top panel), with rolling stream composition (time of day, road type, weather) in the panels below. The scoring snapshot and reference Gaussian are frozen at the end of bootstrap. The shaded band is the seed standard deviation. Vertical dotted lines mark stream-block boundaries with block numbers along the top edge. Acceptance rates are computed over a rolling 500-frame window.

Rolling acceptance rate. Figure 4.2 traces the rolling per-window acceptance rate of the four adaptive variants over the entire post-bootstrap stream alongside the rolling composition panels. Three patterns are visible:

1. **Local responsiveness.** The acceptance rate spikes briefly at most block boundaries, reaching into the 0.5–0.8 range at the sharpest transitions (`city_day_*` → `city_twilight`; `highway_day` → `highway_twi-night`), then tightens again over subsequent refresh cycles as the reference absorbs the new content.

2. **Persistent acceptance.** Even within extended homogeneous segments—including the bootstrap-similar `city_day_cloudy` block and the longer `arterial-urban_day` block—all four adaptive variants maintain an acceptance rate above 10%. This follows from threshold calibration (Section 3.6.2): because τ is set as the $(1-p)$-quantile of in-sample distances, a fraction p of any in-distribution segment is always accepted by construction.

3. **Family separation.** The reservoir variants (red) accept fewer frames than the window variants (orange) on the second half of the stream. By the rural tail, the reservoir holds a broad sample of all prior content and treats most rural frames as redundant, while the recency-biased window has already rotated past highway content and is more permissive.

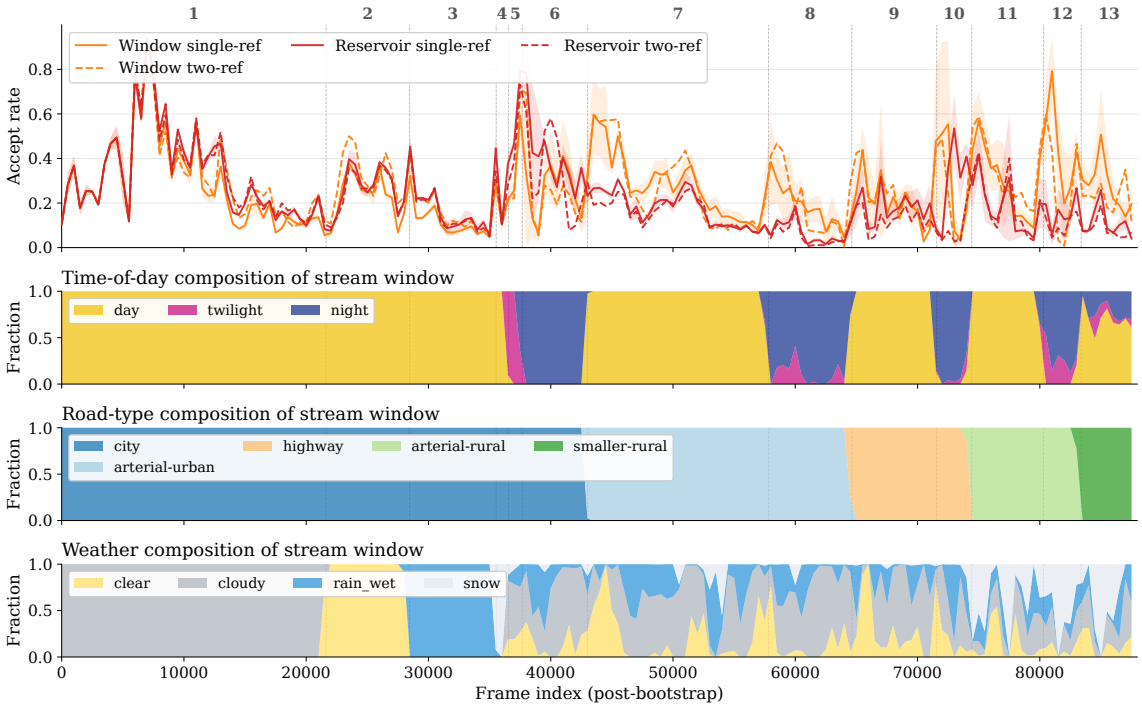

Figure 4.2: Rolling per-window acceptance rate of the four headline adaptive variants on the curated stream (top panel), with rolling stream composition (time of day, road type, weather) in the panels below. Solid lines are seed-mean trajectories with seed-std bands. Dashed lines drop the std band to keep paired single- and two-reference variants visually distinguishable. Block boundaries are marked with dotted vertical lines and block numbers along the top edge. Accept-rate spikes are most pronounced at the twilight and night transitions. The reservoir variants (red) settle below the window variants (orange) on the second half of the stream. Acceptance rates are computed over a rolling 500-frame window.

Per-block acceptance rates. Figure 4.3 shows the mean acceptance rate per stream block for all four adaptive variants, with the static filter on the right-hand axis. In the opening city blocks, all four adaptive variants fall between 0.11 and 0.34. The wet and snow blocks show lower acceptance rates than `city_day_cloudy` (0.30–0.34), likely because those weather blocks arrive after the reference has been updated with city-day content. On the illumination-shift blocks (`city_twilight` and `city_night`), the reservoir family rises to 0.34–0.44 while the window family stays at 0.23–0.29. Further into the stream, the reservoir variants drop to 0.05–0.07 on `arterial-urban_twi-night` as the accumulated reference has absorbed that domain, while the window variants remain near 0.19–0.20. By contrast, the static filter (right axis) saturates near 1.0 on every twi-night and night block, with no reduction as the stream progresses, confirming that the adaptive variants' lower rates reflect calibration rather than reduced domain novelty.

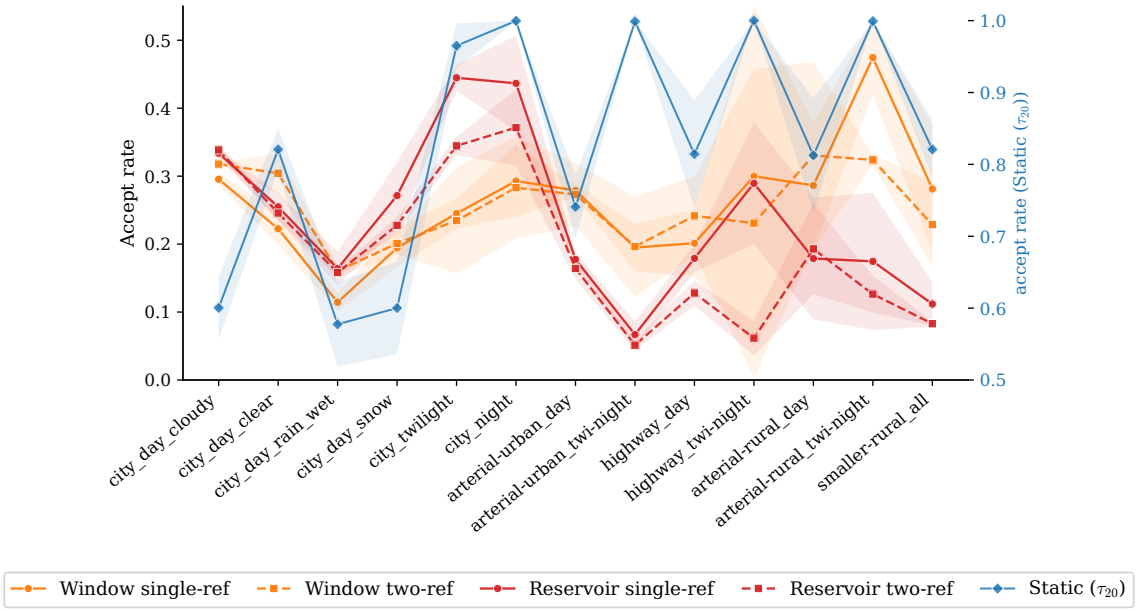

Figure 4.3: Per-block mean acceptance rate for the four headline adaptive variants ($M = R = 1\,500$) and the static filter on the curated stream. Lines join seed-mean acceptance rates over the thirteen curated blocks. Faint shaded bands show the cross-seed standard deviation across three seeds. Window variants are orange (single- and two-reference), reservoir variants are red. The static filter is on the right-hand axis (blue) for scale. Blocks appear in the manifest's streaming order (see also Figure 4.9).

4.1.3 Detection quality at iso-accept

The comparisons below test whether domain-sensitive routing translates into detection gains at matched acceptance rate.

Iso-accept comparison. Figure 4.4 shows smoothed tail-5 validation mAP against empirical acceptance rate for the headline adaptive variants ($M = R = 1\,500$) and the random sweep on the curated stream (Table 4.1). All four headline filters land above their iso-accept random partner, with gaps ranging from +0.0007 for reservoir single-reference to +0.0067 for reservoir two-reference at $\rho = 0.21$. The gains are positive but modest, and must be read alongside the cross-seed variability in the table. Reservoir single-reference's +0.0007 gap is smaller than its random partner's standard deviation ($\sigma \approx 0.0016$) and should be treated as a null result. The other three variants show gaps that exceed the combined seed noise and provide more reliable evidence of a selection benefit. At the block level the gap varies substantially by domain (Table A.1, Appendix A.1), with no single block accounting for the aggregate advantage. At its much higher stream-wide rate of $\rho \approx 0.77$, the static filter falls −0.0033 mAP below `random_p77`, consistent with the acceptance-rate drift documented in Section 4.1.1.

Per-block decomposition. Aggregate mAP can hide per-block variation. Breaking the iso-accept comparison down by block, three of the four headline filters beat

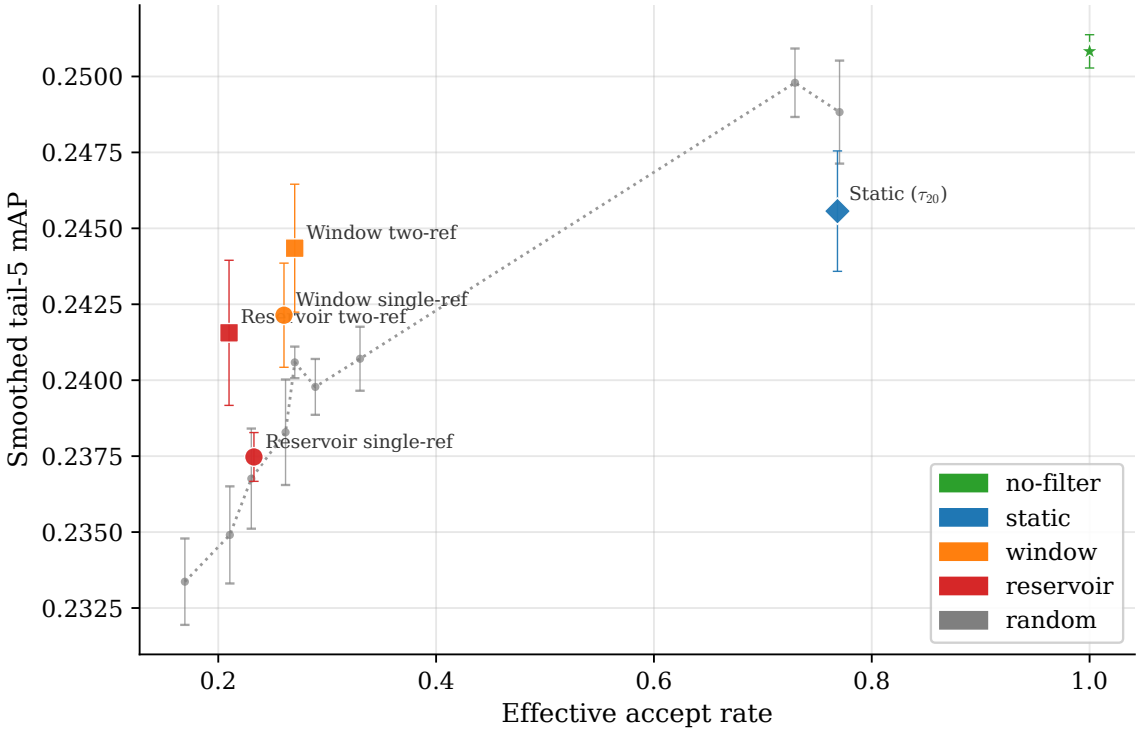

Figure 4.4: Smoothed tail-5 validation mAP vs. empirical acceptance rate for the curated stream. Each marker is a streaming variant. Filter variants appear in color: window family in orange, reservoir family in red, static filter in blue, with the no-filter upper bound at the right edge. The random sweep is a dotted grey envelope; grey error bars at each tested rate show the seed standard deviation. Vertical bars on the colored markers are filter seed standard deviations. The four headline adaptive variants land above their iso-accept random partner at $\rho \approx 0.21$–0.27. The static filter sits below the random envelope at its high acceptance rate.

their random partner on a majority of the twelve blocks: window single-reference on 10/12, window two-reference on 8/12, and reservoir two-reference on 9/12. Reservoir single-reference is the weakest, winning on only 5/12. Block-level gaps span -0.017 to $+0.011$ mAP (Table A.1, Appendix A.1). The win/loss pattern reflects each filter's routing behavior (Figure 4.3) rather than a simple ranking by distance from bootstrap.

Per-category trajectories. Figure 4.5 shows per-category mAP trajectories for reservoir two-reference, its iso-accept random partner, and the no-filter upper bound across the four evaluation categories (`night`, `twilight`, `wet`, `snow`). Reservoir two-reference sits above its random partner on night and twilight through much of the stream, while tracking closely on wet and snow. Both sit below the no-filter upper bound by a roughly constant margin. The window two-reference trajectories (Figure A.1, Appendix A.2) are closer to neutral across these four categories, consistent with the smaller aggregate gap of $+0.0038$ mAP.

Table 4.1: Iso-accept comparison for the four headline adaptive variants ($M = R = 1\,500$, curated stream). Each row is one (filter, iso-accept random) pair. Acceptance rates are empirical means across three seeds; the accept gap is filter minus random. Smoothed mAP is the tail-5 mean for each variant. All four pairs match acceptance rate to within 0.002, so Δ mAP is dominated by selection quality rather than data volume.

Pair	Filter ρ	Random ρ	$\Delta\rho$	Filter mAP	Δ mAP
Window single-ref	0.260	0.262	−0.001	0.2421	+0.0039
Window two-ref	0.270	0.270	0.000	0.2443	+0.0038
Reservoir single-ref	0.233	0.230	+0.002	0.2375	+0.0007
Reservoir two-ref	0.210	0.211	−0.001	0.2416	+0.0067

Table 4.2: Iso-accept comparison for the $M = R = 1\,000$ ablation set on the curated stream. Conventions match Table 4.1; $\Delta\rho$ is filter minus random acceptance rate, Δ mAP is smoothed tail-5 filter minus random. The reservoir no-anchor acceptance gap is larger than the other rows (−0.031) because no random rate was run at $\rho \approx 0.139$. The pairing against `random_p17` is the closest available random partner, and the Δ mAP for this row is therefore partially confounded by the data-volume mismatch.

Pair	Filter ρ	Random ρ	$\Delta\rho$	Filter mAP	Δ mAP
Window single-ref (M=1 000)	0.327	0.330	−0.003	0.2428	+0.0021
Window two-ref (M=1 000)	0.335	0.330	+0.004	0.2458	+0.0051
Window no-anchor (M=1 000)	0.341	0.330	+0.011	0.2386	−0.0021
Reservoir single-ref (R=1 000)	0.203	0.211	−0.008	0.2399	+0.0050
Reservoir two-ref (R=1 000)	0.193	0.211	−0.018	0.2383	+0.0034
Reservoir no-anchor (R=1 000)	0.139	0.169	−0.031	0.2320	−0.0013

4.1.4 Memory size and bootstrap anchor

The headline analysis above fixes the adaptive memory at $M = R = 1\,500$ (matched to the federated per-client budget) and holds the bootstrap anchor on. The ablation set (Table 3.7, $M = R = 1\,000$) tests two design choices at the smaller memory size: the effect of dropping the bootstrap anchor (`*_noBoot`), and the effect of replacing the single-reference Gaussian with the two-reference min rule (`*_twoRef`). Iso-accept comparisons for these variants are listed in Table 4.2.

First, removing the bootstrap anchor has the most consistently negative effect on the iso-accept gap among the ablated design choices. Without it, the reference contracts around recently accepted frames, the threshold tightens, and "novelty" becomes a moving target relative to the recent memory rather than the bootstrap distribution. This mechanism is evidenced in the results: both no-anchor variants are the only adaptive rows with a *negative* iso-accept gap (−0.0021 for window no-anchor, −0.0013 for reservoir no-anchor), and the reservoir no-anchor acceptance rate collapses to $\rho \approx 0.14$, well below the calibrated 0.20 target. Both no-anchor variants are matched to random partners with a small residual acceptance gap (window no-anchor: $\Delta\rho = +0.011$; reservoir no-anchor: $\Delta\rho = -0.031$). The reservoir no-anchor

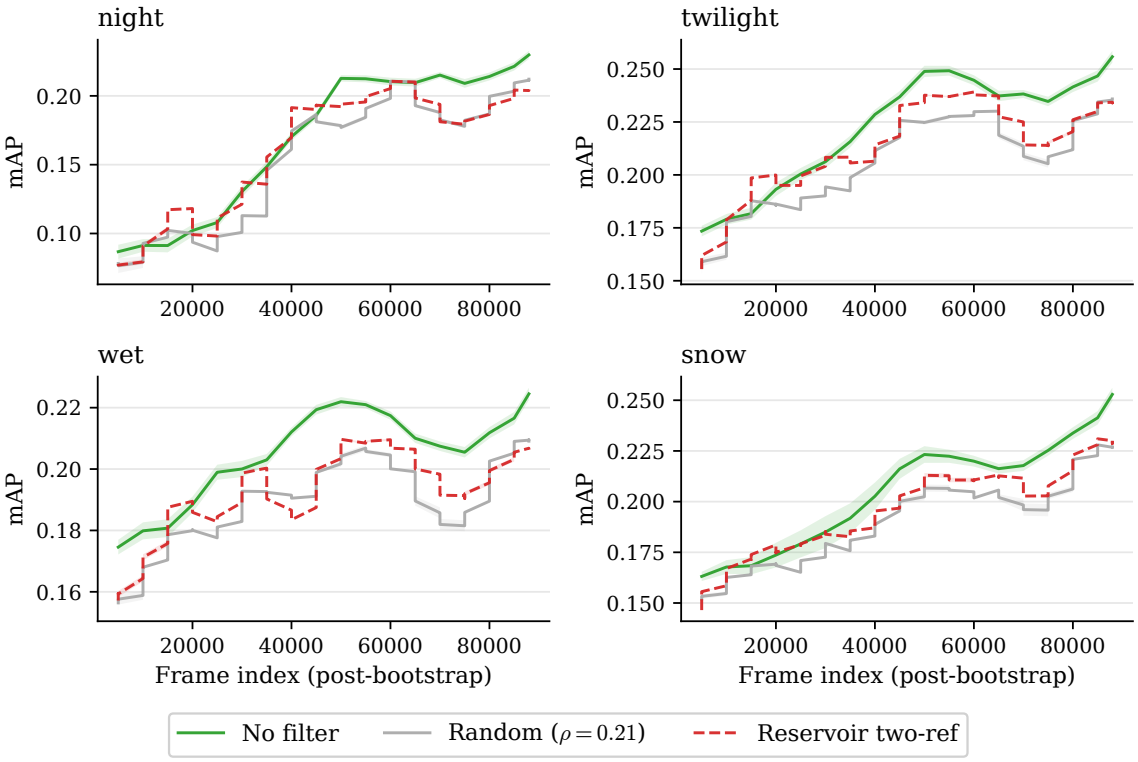

Figure 4.5: Per-category mAP trajectories for reservoir two-reference on the curated stream. The four panels correspond to validation-set subsets obtained by filtering the ∼10 000-frame validation split on metadata: `night` and `twilight` (time-of-day buckets from Section 3.10) and `wet` and `snow` (road-condition buckets). Each curve is the seed-mean mAP at every checkpoint, smoothed with a centered five-checkpoint rolling window. Faint shaded bands are the cross-seed standard deviation across three seeds. Three lines per panel: no-filter upper bound (green), reservoir two-reference (red dashed), and `random_p21` iso-accept random baseline at $\rho = 0.210$ (grey). Reservoir two-reference sits above its random partner on night and twilight through much of the stream, while tracking closely on wet and snow.

gap is the larger concern. Because the swept random baselines do not include a variant at $\rho \approx 0.14$, the closest available partner is `random_p17` at $\rho = 0.169$, and the reported Δ mAP of -0.0013 is partly confounded by data volume. The magnitude of this result should not be compared directly with the other rows. The direction—a negative gap—is interpretable and consistent with the window no-anchor finding.

Second, the two-reference rule and the memory-size choice are secondary. At $M = R = 1\,000$, two-reference improves on single-reference for the window family (+0.0051 vs. +0.0021) but not for the reservoir family (+0.0034 vs. +0.0050), and in both cases the differences sit inside the three-seed standard-deviation band. Reducing memory size from 1 500 to 1 000 frames shifts the iso-accept gap by up to 0.005 mAP but leaves every filter variant above its random partner. Memory size and reference structure are thus secondary to the anchor in this setting.

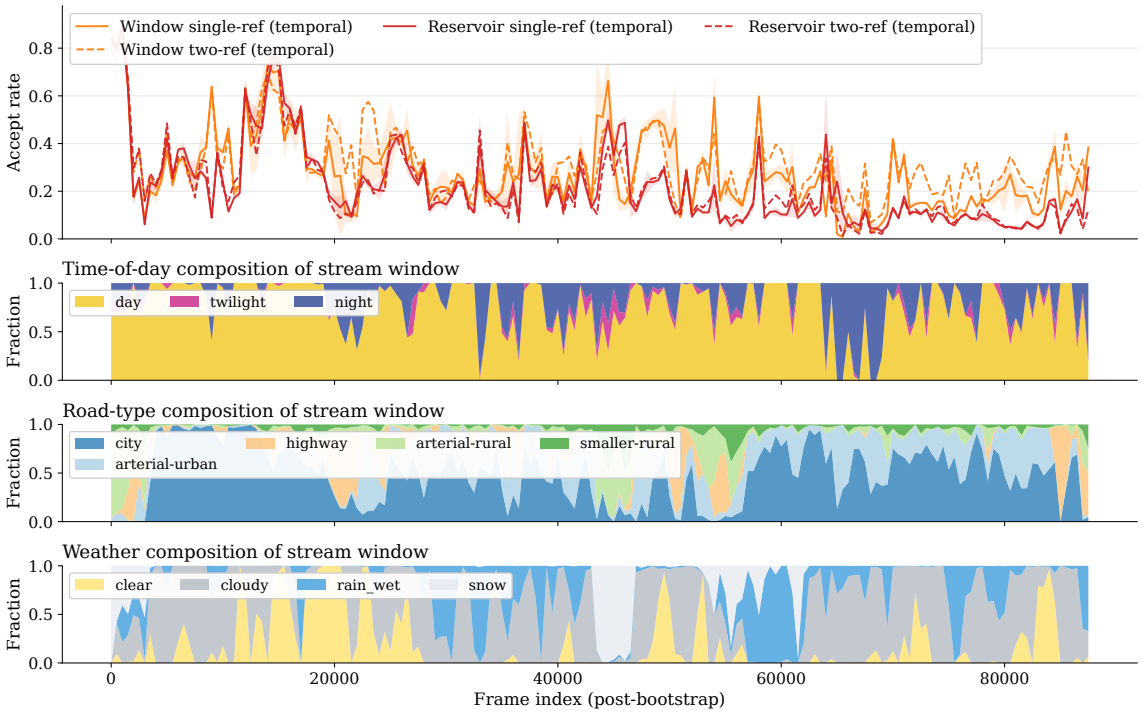

Figure 4.6: Rolling per-window acceptance rate of the headline adaptive variants on the temporal manifest, plotted in the same conventions as Figure 4.2 but without block-boundary annotations (the temporal manifest has no engineered blocks). Composition panels are the same fields (time-of-day, road-type, weather) computed over the same rolling window. Accept-rate spikes are smaller and the rolling composition is a noisy mixture rather than a step function, but the family separation between window and reservoir variants is preserved. Acceptance rates are computed over the same 500-frame rolling window as Figures 4.1 and 4.2.

4.1.5 Temporal replication

The block-ordered curated manifest produces clean step transitions at domain boundaries. A natural concern is whether the filter's behavior depends on those engineered boundaries. Figure 4.6 shows the same analysis on the temporal manifest, where the same post-bootstrap frames are presented in capture-timestamp order (Section 3.4.4). The block boundaries disappear and the rolling composition becomes a noisy mixture, but the filters' qualitative behavior survives: per-window accept rates remain in the 0.15–0.5 range with the same family separation between window and reservoir variants. End-of-stream iso-accept comparisons reproduce the curated-stream result within seed noise (window single-reference +0.0027, window two-reference +0.0014, reservoir single-reference +0.0030, reservoir two-reference +0.0049), all positive but small. The temporal manifest therefore confirms that the results are not an artifact of the engineered block boundaries.

4.1.6 Streaming summary

The streaming results are summarized as follows.

1. The bootstrap-anchored Mahalanobis filter is *domain-sensitive*: per-block acceptance rates vary by a factor of ∼5 across the curated blocks (Figure 4.3), and the static filter's per-window acceptance rate directly reflects the empirical novelty of each segment relative to bootstrap (Figure 4.1).

2. Periodic refresh keeps the filter *calibrated* throughout the stream: the four headline adaptive variants maintain $\rho \approx 0.20$–0.27 (near the $p = 0.20$ target), while the static filter drifts to $\rho \approx 0.77$ (Figures 4.1, 4.2).

3. At iso-accept, novelty-guided selection is *modestly* better than random: the four headline pairs land between +0.0007 and +0.0067 smoothed mAP above their iso-accept random partner (Table 4.1), with the largest gap for reservoir two-reference at the lowest acceptance rate ($\rho = 0.21$).

4. The per-block decomposition shows that novelty-guided selection wins on a *majority* of stream blocks for three of the four headline pairs (window single-reference 10/12, window two-reference 8/12, reservoir two-reference 9/12), but the per-block gaps span −0.017 to +0.011 mAP, so the per-block results are mixed (Table A.1, Appendix A.1).

5. The $M = R = 1\,000$ ablation set (Section 4.1.4, Table 4.2) shows that removing the bootstrap anchor has the most consistently negative effect among the ablated design choices: the two no-anchor variants are the only adaptive rows with a negative iso-accept gap, and the reservoir no-anchor acceptance rate collapses to $\rho \approx 0.14$. Memory size (1 000 vs. 1 500) and the two-reference rule, by contrast, each shift the iso-accept gap by at most 0.005 mAP while keeping every filter variant above its random partner.

6. The temporal manifest reproduces the curated picture within seed noise (Section 4.1.5), so the results are not an artifact of the engineered block structure of the curated manifest.

The Mahalanobis filter's core strength in this setting is domain-shift detection: embedding distances respond reliably to the lighting, weather, and road-type changes that define the stream's domain structure, and the acceptance-rate traces above confirm this sensitivity. One limitation in this setting, however, is that embedding novelty and training utility are not equivalent. A dark night frame with few visible objects can receive a high novelty score yet contribute little to detector learning, whereas a visually familiar city-day frame dense with objects may contribute more. The positive but small iso-accept gap (points 3–4) likely reflects this tension.

4.2 Federated streaming

The federated experiments combine two manifest partitions and two training schedules. Under the curated partition, frames are grouped by domain into four clients (C0–C3); under the temporal partition, the stream is divided into four chronological quartiles (Q1–Q4). The default schedule runs 30 rounds of 1 000 items per client,

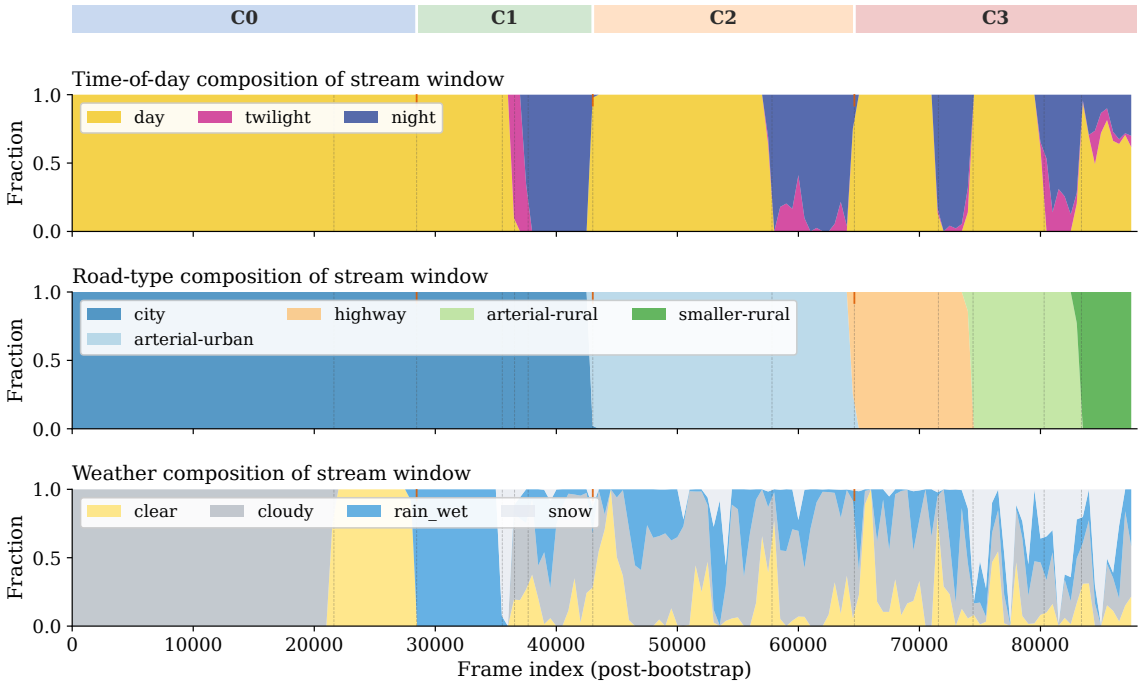

Figure 4.7: Stream-window composition of the curated partition (rolling window of 500 frames) along time of day, road type, and weather. The header strip above the data panels carries one colored ribbon per domain-aligned client, labeled C0–C3. Short orange ticks at the top edge of each composition panel mark the same boundaries. Grey dashed lines mark the underlying stream-block transitions. C0 spans the two city-day blocks (clear and cloudy); C1 covers city adverse weather and low-light (rain, snow, twilight, night); C2 covers arterial-urban blocks; C3 covers highway and rural environments. See Table 3.5 for frame counts.

while the heavyLocal schedule uses 10 rounds of 3 000 items at the same total budget. The main analysis focuses on the curated partition with the default schedule, which provides the strongest per-client domain contrast and the most communication rounds. Results for the other three partition-schedule combinations are reported in Section 4.2.5.

4.2.1 Per-client stream composition

Figure 4.7 shows the stream composition for the four domain-aligned client partitions (C0–C3). Because the partition is domain-aligned, each boundary coincides with a major domain shift. C0 covers the bootstrap-similar blocks (`city_day_clear`, `city_day_cloudy`), consisting entirely of daytime city driving with ∼24% clear weather and the rest cloudy. C1 covers the remaining city blocks (`city_day_rain_wet`, `city_day_snow`, `city_twilight`, `city_night`), bringing ∼37% night frames and ∼65% rain/wet/snow weather. C2 covers the arterial-urban blocks (`arterial-urban_day`, `arterial-urban_twi-night`) and is day-dominated. C3 covers the highway and rural blocks (`highway_day`, `highway_twi-night`, `arterial-rural_day`, `arterial-rural_twi-night`, `smaller-rural_all`), with the most diverse road-type mix (∼42% highway, ∼38% arterial-rural, ∼20% smaller-rural) and ∼25% snow

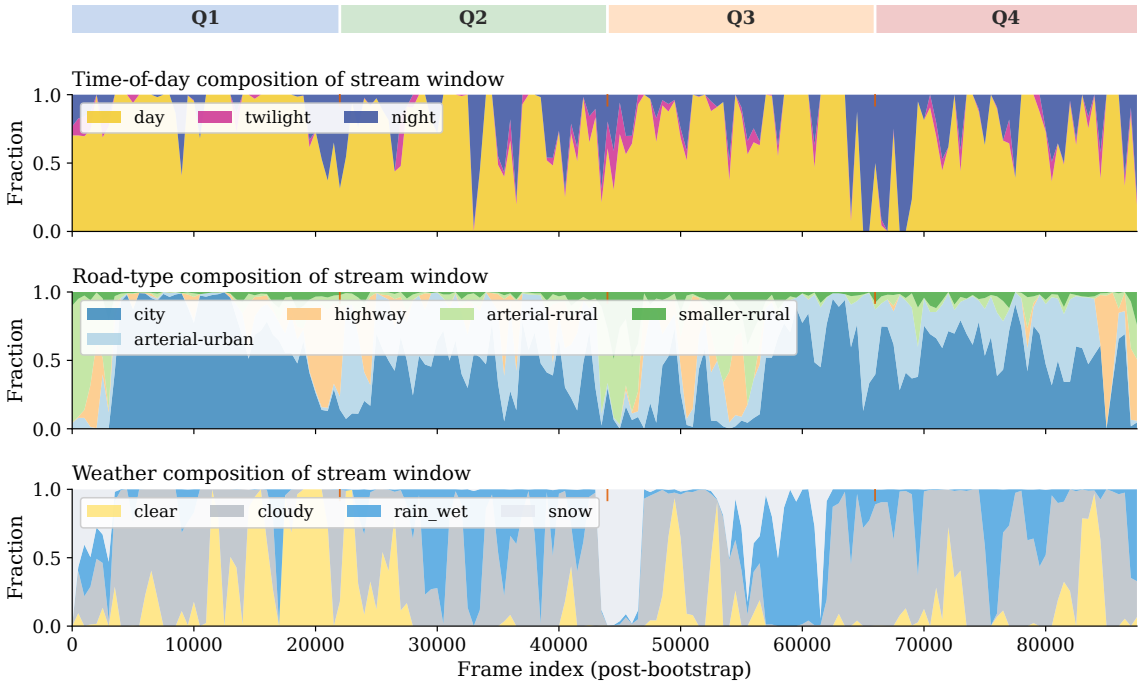

Figure 4.8: Stream-window composition of the temporal partition along the same three dimensions as Figure 4.7 (rolling window of 500 frames). The header strip above the data panels carries one colored ribbon per chronological client, labeled Q1–Q4. Short orange ticks at the top edge of each composition panel mark the same boundaries. Unlike the curated partition, each quartile spans a mixture of road types and conditions, with a chronological rather than domain-aligned gradient. Q4 has roughly three times the night fraction of Q1, and Q3 carries by far the most winter snow content.

weather.

Figure 4.8 shows the temporal partition, where the stream is divided into four equal-sized chronological quartiles (Q1–Q4), each spanning a mixture of road types and conditions. Unlike the curated partition, composition differences between clients are more gradual, following a chronological rather than domain-aligned gradient. Q1 is the most bootstrap-similar quartile (high day fraction, high city share, ∼35% clear weather). Q2 carries the highest rain/wet fraction and the heaviest arterial-urban share. Q3 has by far the most winter snow content (∼24%, vs. ≤8% for the others). Q4 is the most night-heavy (∼30% night frames vs. 10–23% for the others). Because every quartile spans all road types, the temporal partition tests whether the filter can extract a useful selection signal from a mixed-domain stream.

4.2.2 Routing behavior under fleet-pooled refresh

Per-block acceptance rates. Figure 4.9 shows the mean acceptance rate per stream block for the three headline adaptive variants ($M = R = 1\,500$, fleet-pooled), with the static filter on the right-hand axis. The pattern differs markedly from the centralized case. The bootstrap-similar `city_day_cloudy` block shows ac-

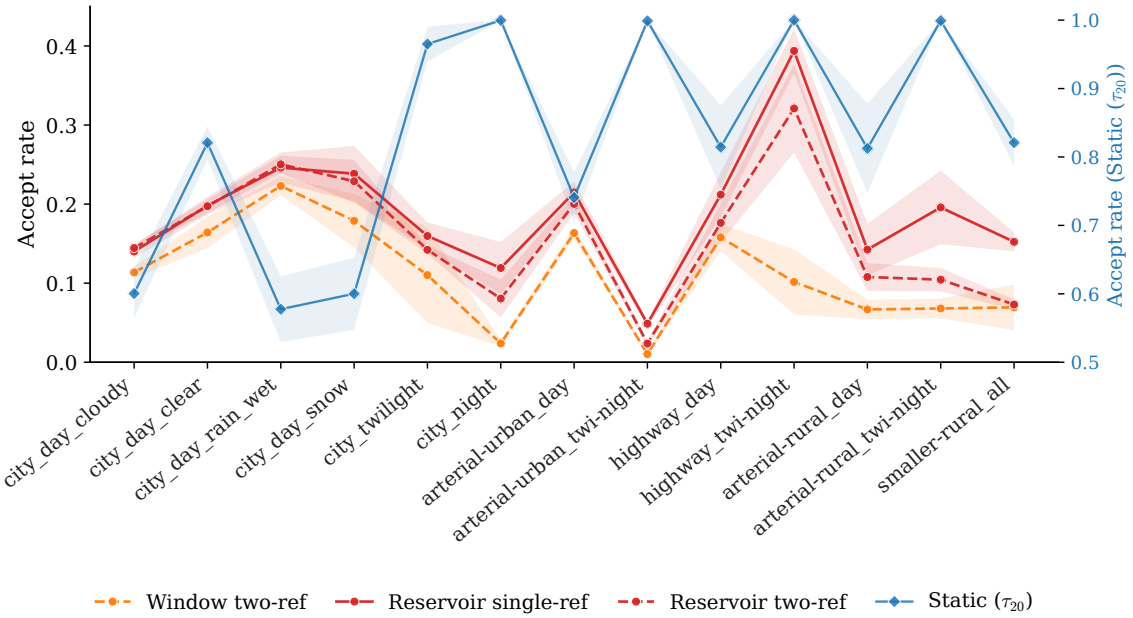

Figure 4.9: Per-block mean acceptance rate for the adaptive variants on the curated partition (default schedule). Lines join seed-mean acceptance rates over the thirteen curated blocks. Faint shaded bands show the cross-seed standard deviation across three seeds. The reservoir variants are red (solid: single-reference; dashed: two-reference), the window two-reference variant is orange, and the static filter is on the right-hand axis (blue) for scale. The acceptance rates at `city_night` and `arterial-urban_twi-night` fall among the lowest adaptive values, while the static filter remains near 1.0 on all twi-night and night blocks. Random baselines are omitted because their per-block acceptance rates are flat by construction.

ceptance rates of 0.11–0.14. Within the C1 city blocks, `city_day_rain_wet` and `city_day_snow` are elevated (0.18–0.25), while `city_twilight` and `city_night` are comparatively low (0.02–0.16), likely because the fleet-pooled reference has accumulated illumination-shifted content from multiple clients before those blocks are fully processed. The sharpest drop is at `arterial-urban_twi-night` (0.01–0.05), where the fleet-pooled reference has most thoroughly absorbed that domain. The reservoir variants then rebound to 0.32–0.39 at `highway_twi-night` while the window family remains near 0.10, suggesting that highway nocturnal content is still largely absent from the city-dominated fleet reference. By contrast, the static filter (right axis) saturates near 1.0 on every twi-night and night block, consistent with the centralized case.

Per-round dynamics. Figure 4.10 shows the per-round per-client acceptance-rate trajectories for the three adaptive variants and the static filter on the curated partition. The per-round trajectories exhibit three notable patterns:

1. **Initial novelty signature.** In the first ∼5 rounds, C1 consistently opens near 0.55–0.60 across all three adaptive variants. C2 opens near the same level, while C3 is more moderate at 0.38–0.45. C0 stays below 0.20 throughout.

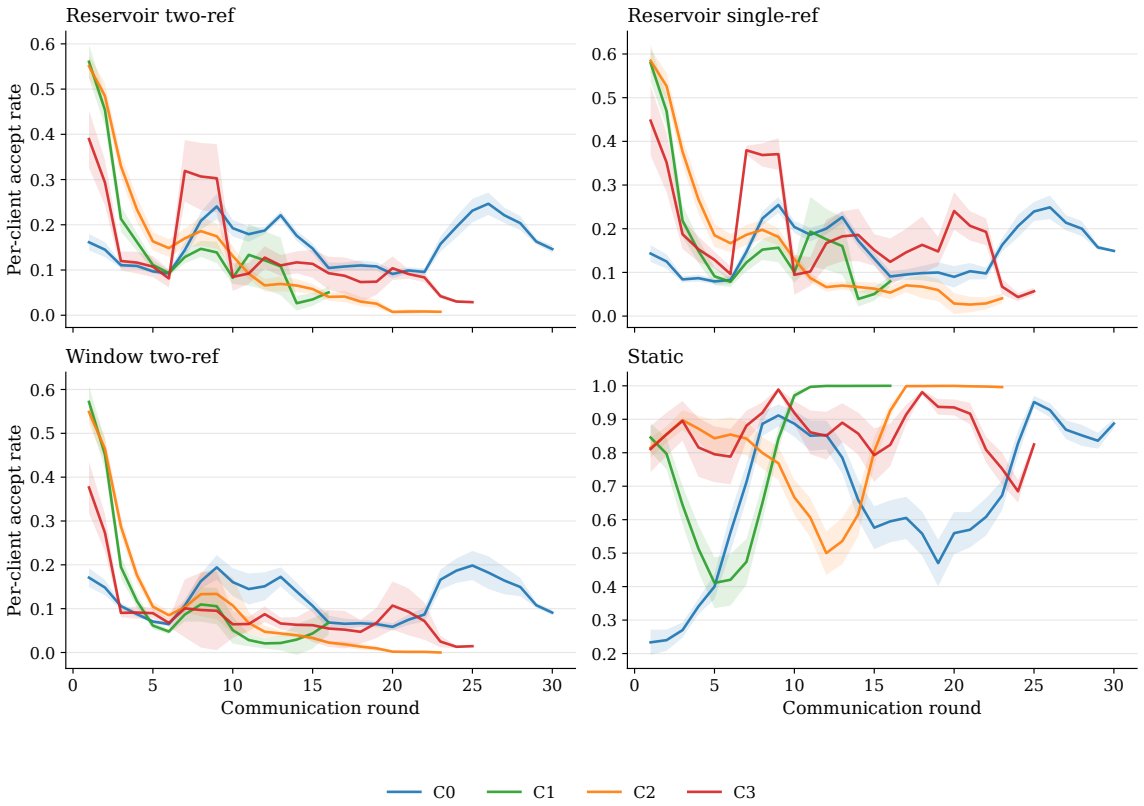

Figure 4.10: Per-round per-client acceptance-rate trajectories for the four filters on the curated partition (default schedule). Each panel is one filter (reservoir two-reference, reservoir single-reference, window two-reference, static filter); each line is one client. Lines are smoothed with a centered three-round window. Shaded bands are the three-seed standard deviation. Y-axes are auto-scaled per panel: the static filter saturates near 1.0 while the adaptive filters operate in 0.0–0.6. All three adaptive filters show a strong early novelty signature (C1–C3 elevated, C0 damped) that converges to a near-uniform 0.10–0.20 acceptance rate by round ∼10 as the fleet-pooled refresh equalizes the global reference. The static filter reaches near-1.0 acceptance rates on C1 and C2 as their night and twi-night frames are processed.

2. **Adaptive convergence.** Within ∼10 rounds, all four clients settle into the calibrated 0.10–0.20 range. The fleet-pooled refresh of Section 3.9.4 equalizes references across clients, so the per-client differentiation diminishes as the global reference accumulates content from all clients.

3. **Static saturation.** The static filter shows high acceptance rates throughout. Without refresh, the acceptance rate approaches 1.0 on segments that are domain-shifted relative to the bootstrap, as the filter accepts nearly every frame on night and twi-night content.

Per-round dynamics on the temporal partition. Figure 4.11 shows the same analysis for the reservoir two-reference filter on the temporal manifest. The chronological partition produces a different early signature. Q2 is the most elevated client in the first ∼5 rounds, consistent with the rain/wet fraction visible in Figure 4.8,

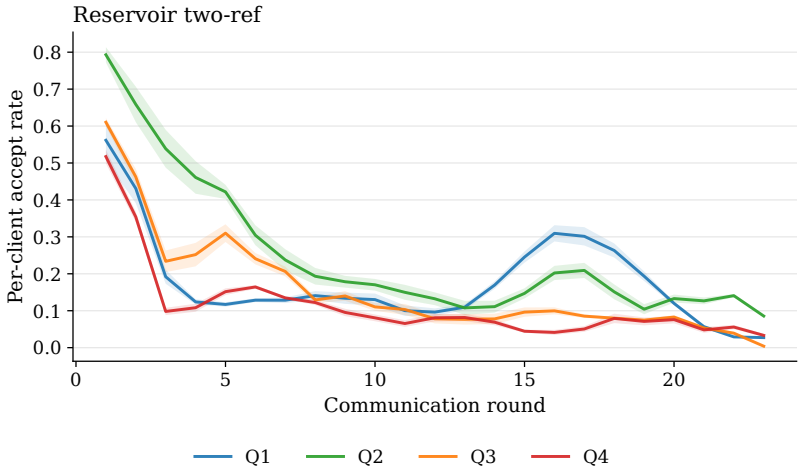

Figure 4.11: Per-round per-client acceptance-rate trajectories for the reservoir two-reference filter on the temporal partition (default schedule). Conventions follow Figure 4.10. Only one panel is shown because only the reservoir two-reference filter is run as an adaptive variant on the temporal partition (Table 3.8). In the first ∼5 rounds, Q2 is the most elevated client. Q1 starts at a comparatively low acceptance rate, consistent with its bootstrap-similar city-day content.

while Q1 starts close to C0's curated trajectory. The mid-round bump on Q1 around round 15 likely arises because the fleet-pooled reference, updated with content from Q2–Q4, has drifted away from Q1's bootstrap-similar frames. Convergence to a near-uniform ∼0.10 acceptance rate is slower than in the curated partition, as temporal clients span a mix of conditions rather than one dominant domain.

Per-client novelty ratio. Table 4.3 summarizes the per-client routing as the *novelty ratio*: mean acceptance rate on C1–C3 divided by the fleet-wide acceptance rate. A value above 1.0 indicates that C1–C3 each accept frames at a higher rate than the fleet-wide average, implying C0 accepts below average. Since C0 is the most bootstrap-familiar client rather than the bootstrap itself, this ratio is a partition-specific indicator of routing direction rather than an absolute novelty measure. The per-round accept-rate trajectories in Figure 4.10 break this down by individual client. The static filter is the clearest router (novelty ratio 1.249 on the curated partition) but saturates and drifts. Among the adaptive filters, the single-reference variants route in the same direction: reservoir single-reference and window single-reference both reach 1.207, meaning C1–C3 contribute about 20% more accepted frames than their fleet share. The two-reference variants are closer to neutral (window two-reference at 0.933, reservoir two-reference at 0.997), likely because the min scoring rule folds recent client data into the running reference more aggressively, reducing the per-client novelty signal.

4.2.3 Detection quality at iso-accept

Iso-accept comparison. Figure 4.12 shows smoothed tail-5 validation mAP against empirical acceptance rate for the curated partition (default schedule). At $\rho \approx 0.12$–

Table 4.3: Per-client *novelty ratio* (mean acceptance rate on C1–C3 divided by fleet-wide acceptance rate) for the adaptive filters across all four partition-schedule combinations. Random partners are omitted (they sit on ∼1.00 by construction). Shaded rows mark the headline reservoir two-reference filter in each combination.

Filter	Setting	$\overline{\rho}$	Nov. ratio
Static τ_{20}	default, curated	0.769	1.249
Window single-ref	default, curated	0.117	1.207
Window two-ref	default, curated	0.118	0.933
Reservoir single-ref	default, curated	0.175	1.207
Reservoir two-ref	default, curated	0.155	0.997
Reservoir two-ref	default, temporal	0.186	0.974
Window single-ref	heavyLocal, curated	0.269	1.253
Window two-ref	heavyLocal, curated	0.251	1.200
Reservoir single-ref	heavyLocal, curated	0.267	1.254
Reservoir two-ref	heavyLocal, curated	0.249	1.222
Reservoir two-ref	heavyLocal, temporal	0.297	1.045

Table 4.4: Iso-accept comparison for the adaptive filters on the curated partition (default schedule). Each row is one (filter, iso-accept random) pair; pairings follow the Partner column of Table 3.8. Acceptance rates are empirical means across three seeds; the accept gap is filter minus random. Smoothed mAP is the tail-5 mean for each variant.

Pair	Filter ρ	Random ρ	$\Delta\rho$	Filter mAP	Δ mAP
Static τ_{20}	0.769	0.768	+0.000	0.2312	−0.0022
Window single-ref	0.117	0.120	−0.003	0.2242	+0.0005
Window two-ref	0.118	0.120	−0.001	0.2245	+0.0008
Reservoir single-ref	0.175	0.179	−0.004	0.2265	+0.0016
Reservoir two-ref	0.155	0.149	+0.005	0.2259	+0.0049

0.18, all four adaptive filters land above their iso-accept random partner, from +0.0005 to +0.0049 mAP. The largest gap is for reservoir two-reference (+0.0049 at $\rho = 0.155$), the same family as in the centralized setting. The static filter sits below its iso-accept random partner at $\rho \approx 0.77$ (−0.0022) because without refresh it accepts nearly every frame on night and twi-night blocks. The four headline filter acceptance rates match their random partners to within 0.006, so the small gaps measure selection quality rather than data volume (Table 4.4).

Per-category trajectories. Figure 4.13 shows per-category mAP trajectories for reservoir two-reference, its iso-accept random partner, and the no-filter upper bound across the four evaluation categories (`night`, `twilight`, `wet`, `snow`). Validation mAP rises through rounds 5–20 and then dips in later rounds. The same dip appears in the no-filter upper bound. Reservoir two-reference opens above its random partner in the first ∼10 rounds, with the early advantage fading over subsequent rounds. The window two-reference trajectories (Figure A.2, Appendix A.3) show the same early lead but diverge thereafter: the filter falls clearly below random on night from

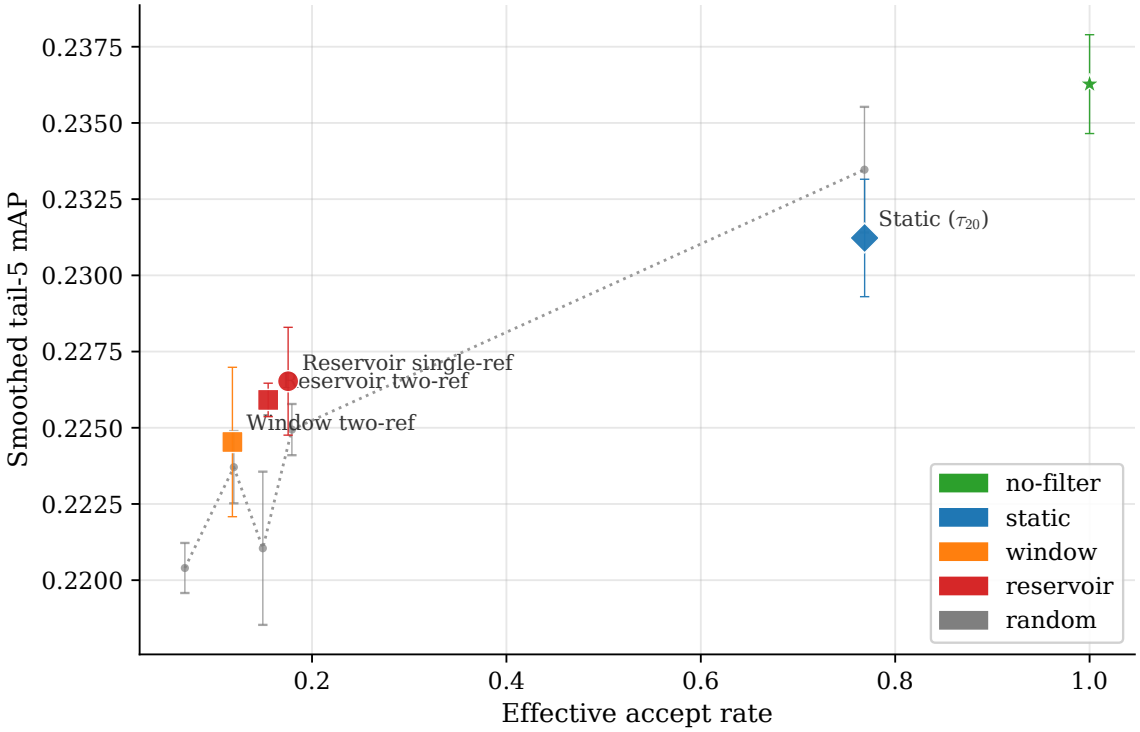

Figure 4.12: Smoothed tail-5 validation mAP vs. empirical acceptance rate for the curated partition (default schedule). Each marker is a federated variant. Filter variants appear in color: window family in orange, reservoir family in red, static filter in blue, with the no-filter upper bound at the right edge. The random sweep is a dotted grey envelope; grey error bars at each tested rate show the seed standard deviation. Both reservoir variants sit above the random envelope at $\rho \approx 0.12$–0.18; the window variants exceed the mean random partner but remain within seed noise. The static filter sits below the envelope at $\rho \approx 0.77$.

round ∼10 onward while remaining near-neutral on wet and snow.

4.2.4 Sensitivity ablations

Two design choices are evaluated as ablations: the calibration percentile (Section 3.6.2) and the two-reference rule (Section 3.8.2). Each ablated variant is compared to a matched baseline within the same partition-schedule combination, with random partners matched to the ablated filter's empirical acceptance rate (Table 3.8). The server-side refresh cadence is not ablated here. The centralized static filter (Section 4.1.1) already demonstrates the same drift mechanism more cleanly and motivates the once-per-round schedule used here.

Calibration percentile. Table 4.5 tests whether tightening the calibration percentile from $p = 0.20$ to $p = 0.10$ (or $p = 0.15$ under the heavyLocal schedule) can save compute without hurting detection quality. Every comparison is negative: the tighter budget reduces mAP by 0.006–0.011 on the default schedule and by 0.003–0.008 on heavyLocal. Halving the acceptance rate roughly halves the num-

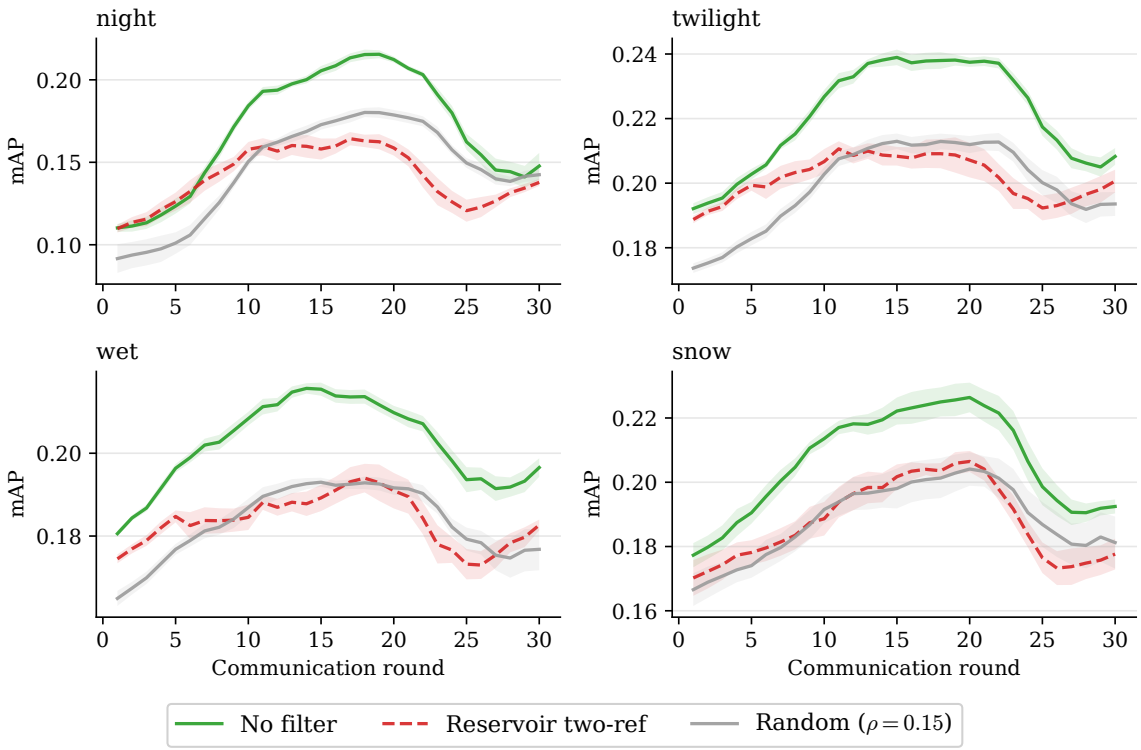

Figure 4.13: Per-category mAP trajectories for reservoir two-reference on the curated partition (default schedule). The four panels correspond to validation-set subsets filtered on metadata (Section 3.10): `night` and `twilight` (time-of-day buckets) and `wet` and `snow` (road-condition buckets). Each curve is the seed-mean mAP at every communication round, smoothed with a centered five-round window. Faint shaded bands are the cross-seed standard deviation across three seeds. Three lines per panel: no-filter upper bound (green), reservoir two-reference (red dashed), and `fed_random_p15` iso-accept random baseline (grey). Reservoir two-reference opens above its random partner in the first ∼10 rounds; the early advantage then fades over subsequent rounds.

ber of gradient steps, from ∼2 700 to ∼1 300 on the default schedule (Table 3.8). The buffer-bounded optimizer (Section 2.6) cannot recover the lost training volume through smarter selection alone. Tightening the calibration percentile therefore does not offer a practical compute saving in the federated setting, where per-client gradient steps are already fewer than in the centralized case.

Two-reference rule. At matched percentile, replacing the single-reference Gaussian with the two-reference min of Section 3.8.2 is roughly neutral on the default schedule (+0.0003 window, −0.0006 reservoir, +0.0035 reservoir at τ_{10}) but clearly positive under the heavyLocal schedule (+0.0120 for window, +0.0011 for reservoir, +0.0036 for reservoir τ_{15}; Table 4.6). One plausible explanation, consistent with the design rationale in Section 3.8.2, is that the two-reference rule is most beneficial when the accepted memory has drifted sufficiently far from the bootstrap distribution that a single Gaussian provides a poor fit to the combined reference set. That condition is met sooner under heavyLocal, where each client trains on a longer

Table 4.5: Federated ablation: calibration percentile. Each row compares a τ_{20} baseline against a tighter τ_{10} (or τ_{15}) variant within the same partition-schedule combination. Δ mAP is the ablated minus the baseline smoothed tail-5 mAP; $\Delta\rho$ is the ablated minus the baseline empirical acceptance rate (negative because the tighter percentile accepts fewer frames).

Pair	Setting	$\Delta\rho$	Baseline mAP	Δ mAP
Window: $\tau_{20}\to\tau_{10}$	default, curated	−0.062	0.2242	−0.0057
Reservoir: $\tau_{20}\to\tau_{10}$	default, curated	−0.099	0.2265	−0.0105
Reservoir two-ref: $\tau_{20}\to\tau_{10}$	default, curated	−0.084	0.2259	−0.0063
Reservoir: $\tau_{20}\to\tau_{15}$	heavyLocal, curated	−0.050	0.2319	−0.0053
Reservoir two-ref: $\tau_{20}\to\tau_{15}$	heavyLocal, curated	−0.042	0.2329	−0.0028
Reservoir two-ref: $\tau_{20}\to\tau_{10}$	heavyLocal, curated	−0.086	0.2329	−0.0078

Table 4.6: Federated ablation: two-reference vs. single-reference at matched percentile. Δ mAP is two-reference minus single-reference smoothed tail-5 mAP within the same variant family, percentile, and partition-schedule combination.

Pair	Setting	Single-ref mAP	Δ mAP
Window	default, curated	0.2242	+0.0003
Reservoir	default, curated	0.2265	−0.0006
Reservoir (τ_{10})	default, curated	0.2161	+0.0035
Window (heavy-local)	heavyLocal, curated	0.2136	+0.0120
Reservoir (heavy-local)	heavyLocal, curated	0.2319	+0.0011
Reservoir (heavy-local, τ_{15})	heavyLocal, curated	0.2266	+0.0036

sub-stream before the next refresh.

4.2.5 Temporal and heavyLocal replication

The curated partition with the default schedule provides the most favorable conditions for domain-sensitive routing, given its strong per-client domain contrast and high number of communication rounds. The three remaining partition-schedule combinations test whether the main result holds under weaker per-client domain structure and fewer communication rounds.

Temporal manifest, default schedule. With the chronological partition (Q1–Q4), the two reservoir two-reference budgets land within ±0.001 of their iso-accept random partner: −0.0010 at $p = 0.15$ ($\rho = 0.146$) and +0.0011 at $p = 0.20$ ($\rho = 0.186$). The sign of the gap remains the same as the curated partition (the $p = 0.20$ filter beats random while the lower-budget $p = 0.15$ filter does not), but the absolute gap shrinks from +0.0049 to +0.0011 without per-client domain contrast.

Curated manifest, heavyLocal schedule. The no-filter upper bound is essentially unchanged from the default schedule (0.2360 vs. 0.2363). Iso-accept gaps against the $\rho = 0.26$ random partner are window single-reference −0.019, window two-reference −0.007, reservoir single-reference −0.001, and reservoir two-reference

+0.000 (Table 3.8). The window single-reference loss of −0.019 exceeds the three-seed standard deviation and is the largest negative gap in the chapter. Reservoir two-reference is the only headline variant with a non-negative iso-accept gap, consistent with the default-schedule result. The window family's larger losses are consistent with the recency-biased window being more sensitive to longer per-round training episodes: each client exhausts its local data in fewer rounds, and by the time the server refreshes, the window reference has drifted further from current client content than the reservoir reference does. Under longer local episodes, the window memory retains exclusively recent content, losing all representation of earlier stream segments by refresh time—which means the fleet reference is assembled from a narrower slice of each client's data and the bootstrap anchor's stabilizing role is diminished.

Temporal manifest, heavyLocal schedule. This is the only combination where the filter clearly beats its iso-accept random at both surveyed budgets: +0.0026 at $p = 0.15$ and +0.0037 at $p = 0.20$. Two possible explanations may contribute, though the data do not separate them. First, Q4 carries the highest night-frame share of any quartile (∼29%). The fleet reference pools content from all four clients, so Q4's night-heavy frames remain relatively novel to the scoring model throughout training, giving the filter a sustained novelty signal to exploit. Second, the heavyLocal schedule's longer per-round episodes may amplify per-client selection differences that are too small to detect under the default schedule. The gap (+0.0037 at $\rho \approx 0.30$) is nonetheless smaller than in the main curated result (+0.0049 at $\rho = 0.155$), indicating the per-frame selection advantage is weaker in this mixed-domain setting.

4.2.6 Federated summary

The federated results are summarized as follows.

1. The Mahalanobis filter is *domain-sensitive* in the federated setting. The static filter's per-block acceptance rates follow the centralized pattern (high on twilight and night blocks). All three adaptive variants show the same bootstrap-anchored novelty signature in their first ∼5 rounds: C1–C3 elevated, C0 damped (Figure 4.10).

2. Fleet-pooled refresh keeps the filter *calibrated*. The adaptive variants converge to a near-uniform 0.10–0.20 acceptance rate by round ∼10, while the static filter's acceptance rate approaches 1.0 on blocks that are domain-shifted relative to the bootstrap. The single-reference adaptive variants retain a residual novelty ratio of ∼1.2 on the curated partition, while the two-reference variants sit near 1.0 (Table 4.3).

3. At iso-accept, all four adaptive filters land *above* their random partner on the curated partition, from +0.0005 to +0.0049 mAP. The largest gap is for reservoir two-reference (+0.0049 at $\rho = 0.155$, Table 4.4), replicating the centralized result. Per-category trajectories show reservoir two-reference opening

above its random partner in the early rounds, with the early advantage fading over subsequent rounds (Figure 4.13), while window two-reference develops a clear night deficit after the early rounds (Figure A.2, Appendix A.3).

4. Results replicate across all four partition-schedule combinations. Reservoir two-reference is the only adaptive variant with a non-negative iso-accept gap in every combination. Among the three replication settings, temporal heavy-Local produces the largest positive gap (+0.0037 at $p = 0.20$). Under the default schedule, the higher number of communication rounds pushes per-client novelty routing toward 1.0.

5. Sensitivity ablations isolate two secondary axes. Tightening the calibration percentile from $p = 0.20$ to $p = 0.10$ reduces mAP in every combination (Table 4.5): federated clients cannot recover the missing gradient steps through smarter selection. The two-reference rule is neutral on the default schedule but improves by up to +0.0120 mAP on heavyLocal (Table 4.6), where the accepted memory drifts furthest from the bootstrap and a single Gaussian is the poorest fit.

5

Conclusion

This thesis studied novelty-aware frame selection for streaming object detection. A Mahalanobis-distance filter scores each incoming frame against a bootstrap-anchored reference Gaussian, accepting frames whose score exceeds a calibrated threshold for training. The scoring snapshot, reference, and threshold are periodically refreshed as the detector adapts. The mechanism was evaluated centrally and in a four-client federated setting, across two stream orderings and multiple design variants, with volume-matched random baselines throughout.

5.1 Findings

The five research questions posed in Section 1.2 are addressed in turn below.

Domain sensitivity and calibration (RQ1, RQ2). The distribution-based filter is genuinely domain-sensitive (RQ1): per-block acceptance rates vary by a factor of $\sim$5 across the curated stream, with the lowest values on city-day blocks close to the bootstrap domain and elevated values on twilight and night blocks (Figure 4.3). Calibration at the $p = 0.20$ target, however, requires periodic refresh (RQ2). Without it, the static filter drifts from its calibrated target to an empirical $\rho \approx 0.77$ by the end of the stream, approaching $\rho = 1.0$ on twi-night and night blocks (Figure 4.1). Periodic refresh jointly recalibrates the snapshot, reference Gaussian, and threshold at each interval, maintaining per-block sensitivity while keeping the empirical acceptance rate near the calibrated target throughout the stream.

Design choices and detection quality (RQ3, RQ4). Among the adaptive design choices (RQ3), removing the bootstrap anchor has the most consistently negative effect, producing the only adaptive variants with a negative iso-accept gap and causing the reservoir no-anchor acceptance rate to collapse to $\rho \approx 0.14$ as novelty drifts toward a moving target. The window-versus-reservoir difference is secondary but visible in routing behavior: by the rural tail the reservoir treats late-stream frames as largely redundant while the window remains more permissive, reflecting their different priors about what to remember.

Evaluated against volume-matched random baselines (RQ4), three of the four headline adaptive variants show iso-accept gaps that exceed the cross-seed noise on the curated stream (Table 4.1): window single-reference (+0.0039), window two-reference

(+0.0038), and reservoir two-reference (+0.0067). The smallest gap, reservoir single-reference at +0.0007, is within the standard deviation of its random partner and should be treated as a null result (Section 3.10.5). Both the routing pattern and the reliable iso-accept gaps replicate on the temporal manifest, confirming these results are not artifacts of the engineered block structure.

Although consistently positive, the gains are modest: distributional novelty and training utility do not fully align. A night frame with few visible objects scores high under the bootstrap Gaussian yet contributes little to detector learning, whereas a city-day frame dense with annotations may contribute more gradient signal per training step.

Federated extension (RQ5). The proposed mechanism extends naturally to the four-client federated setting (RQ5), broadly replicating the routing and calibration behavior of the centralized case. In the first ∼5 rounds, C1–C3 show elevated acceptance rates while C0 remains near the calibrated target, reflecting each client's distance from the bootstrap domain—the same novelty signature seen in the streaming setting. Fleet-pooled refresh then causes per-client rates to converge by round ∼10, as the global reference expands to span all clients' data (Figure 4.10). At iso-accept, the four adaptive headline filters land between +0.0005 and +0.0049 smoothed mAP above their random partners on the curated partition (Table 4.4). With a shared fleet reference, differences in per-client acceptance rates reflect genuine differences in data composition rather than drift in each client's separately maintained reference, keeping the routing patterns interpretable across clients. Across both settings, reservoir two-reference is the only variant with a non-negative iso-accept gap in every partition-schedule combination tested—the strongest practical recommendation this work can offer.

5.2 Limitations and future work

The modest detection gains partly reflect the streaming training setup: a small buffer limits gradient steps per accepted frame, and accepting only a fraction of frames means less data overall than a no-filter baseline. Within the iso-accept comparison, filter and random operate under the same constraints, so these factors limit the absolute scale of gains without biasing the comparison. A more fundamental limitation is that distributional novelty does not fully align with training utility. This misalignment would persist even with a larger buffer or a higher acceptance rate, because it reflects what the novelty score measures rather than how many frames are accepted.

All experiments assume that labels are available for every accepted frame. In practice, annotations would require human labelers, model-generated pseudo-labels, or a self-supervised objective. Given that the framework is designed to reduce the number of frames forwarded for training, this assumption is significant: the novelty filter reduces training compute but does not reduce annotation cost unless combined with a labeling strategy. How annotation quality and availability interact

with novelty-guided selection is therefore an important open question with direct practical implications.

Closing the gap between distributional novelty and training utility is the most promising direction for future work. The most direct path is extending the novelty score with detection-relevant signals such as object density or per-frame detector confidence. ZOD is described as a multimodal dataset [3] and provides additional sensor streams beyond images, including LiDAR point clouds. Incorporating such modalities into the backbone representation could further enrich the novelty signal and better capture the full domain shift experienced by the vehicle.

The empirical findings also suggest directions for improving the scoring model and memory structure. The dominant role of the bootstrap anchor suggests that allowing it to adapt as the target distribution evolves could improve calibration under large domain shifts. More generally, replacing the single Gaussian reference with a mixture model that represents multiple driving domains in the feature space could yield a more expressive novelty signal than the two-reference ablation tested here. The choice between sliding-window and reservoir memory likewise depends on application requirements: the reservoir maintains broader coverage of all seen content but may under-weight frames that are genuinely novel yet similar to accumulated memory, while the window stays current at the cost of forgetting earlier domains.

A separate practical constraint in the federated design is that the server re-embeds accepted frames under the current global model at every round. In real deployment this would require transmitting raw frame data from clients to the server, raising bandwidth and privacy concerns at fleet scale. Approaches that avoid centralizing raw frames, such as computing embeddings locally under a shared model snapshot, would be a prerequisite for privacy-preserving deployment. This requires that clients use a synchronized snapshot at each scoring round.

As training progresses through later stream domains, detection quality on earlier-encountered domains erodes. Replay or regularization-based continual learning could help mitigate this forgetting. The evaluation is further limited to ZOD Frames, a sparse-keyframe dataset in which consecutive training frames typically originate from different drives. Whether comparable gains hold on continuous high-rate streams, where the primary challenge shifts to filtering near-identical frames rather than detecting domain shift, remains an open question.

Bibliography

[1] G. I. Parisi, R. Kemker, J. L. Part, C. Kanan, and S. Wermter, "Continual lifelong learning with neural networks: A review," *Neural Networks*, vol. 113, pp. 54–71, 2019.

[2] K. Lee, K. Lee, H. Lee, and J. Shin, "A simple unified framework for detecting out-of-distribution samples and adversarial attacks," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2018, pp. 7167–7177.

[3] M. Alibeigi, W. Ljungbergh, A. Tonderski, G. Hess, A. Lilja, C. Lindström, D. Motorniuk, J. Fu, J. Widahl, and C. Petersson, "Zenseact open dataset: A large-scale and diverse multimodal dataset for autonomous driving," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023, pp. 20 178–20 188.

[4] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2017, pp. 1273–1282.

[5] Z. Tian, C. Shen, H. Chen, and T. He, "FCOS: Fully convolutional one-stage object detection," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019, pp. 9627–9636.

[6] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 2117–2125.

[7] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778.

[8] 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.

[9] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: Common objects in context," in *European Conference on Computer Vision (ECCV)*. Springer, 2014, pp. 740–755.

[10] B. Settles, *Active Learning*, ser. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool, 2012.

[11] B. Mirzasoleiman, J. Bilmes, and J. Leskovec, "Coresets for data-efficient training of machine learning models," in *Proceedings of the International Conference on Machine Learning (ICML)*, 2020, pp. 6950–6960.

[12] M. Paul, S. Ganguli, and G. K. Dziugaite, "Deep learning on a data diet: Finding important examples early in training," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2021, pp. 20 596–20 607.

[13] P. C. Mahalanobis, "On the generalized distance in statistics," *Proceedings of the National Institute of Sciences of India*, vol. 2, no. 1, pp. 49–55, 1936.

[14] D. Hendrycks and K. Gimpel, "A baseline for detecting misclassified and out-of-distribution examples in neural networks," in *International Conference on Learning Representations (ICLR)*, 2017.

[15] J. S. Vitter, "Random sampling with a reservoir," *ACM Transactions on Mathematical Software*, vol. 11, no. 1, pp. 37–57, 1985.

[16] J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, and A. Bouchachia, "A survey on concept drift adaptation," *ACM Computing Surveys*, vol. 46, no. 4, pp. 44:1–44:37, 2014.

[17] P. Kairouz, H. B. McMahan, B. Avent, A. Bellet, M. Bennis, A. N. Bhagoji, K. Bonawitz, Z. Charles, G. Cormode, R. Cummings, R. G. L. D'Oliveira, H. Eichner, S. E. Rouayheb, D. Evans, J. Gardner, Z. Garrett, A. Gascón, B. Ghazi, P. B. Gibbons, M. Gruteser, Z. Harchaoui, C. He, L. He, Z. Huo, B. Hutchinson, J. Hsu, M. Jaggi, T. Javidi, G. Joshi, M. Khodak, J. Konečný, A. Korolova, F. Koushanfar, S. Koyejo, T. Lepoint, Y. Liu, P. Mittal, M. Mohri, R. Nock, A. Özgür, R. Pagh, H. Qi, D. Ramage, R. Raskar, M. Raykova, D. Song, W. Song, S. U. Stich, Z. Sun, A. T. Suresh, F. Tramèr, P. Vepakomma, J. Wang, L. Xiong, Z. Xu, Q. Yang, F. X. Yu, H. Yu, and S. Zhao, "Advances and open problems in federated learning," *Foundations and Trends in Machine Learning*, vol. 14, no. 1–2, pp. 1–210, 2021.

[18] Y. Zhao, M. Li, L. Lai, N. Suda, D. Civin, and V. Chandra, "Federated learning with non-IID data," *arXiv preprint arXiv:1806.00582*, 2018.

[19] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated optimization in heterogeneous networks," in *Proceedings of Machine Learning and Systems (MLSys)*, 2020, pp. 429–450.

[20] S. P. Karimireddy, S. Kale, M. Mohri, S. J. Reddi, S. U. Stich, and A. T. Suresh, "SCAFFOLD: Stochastic controlled averaging for federated learning," in *Proceedings of the International Conference on Machine Learning (ICML)*, 2020, pp. 5132–5143.

[21] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, D. Hassabis,

C. Clopath, D. Kumaran, and R. Hadsell, “Overcoming catastrophic forgetting in neural networks,” *Proceedings of the National Academy of Sciences*, vol. 114, no. 13, pp. 3521–3526, 2017.

[22] P. Reis, J. Ransiek, D. Petri, T. Schürmann, J. Langner, and E. Sax, “A data-driven novelty score for diverse in-vehicle data recording,” in *28th IEEE International Conference on Intelligent Transportation Systems (ITSC)*, 2025, pp. 248–255.

[23] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, and L. Fei-Fei, “ImageNet large scale visual recognition challenge,” *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, 2015.

A

Supplementary results

A.1 Streaming: per-block iso-accept decomposition

Table A.1 reports the per-block iso-accept Δ mAP for the four headline adaptive variants on the curated stream ($M = R = 1\,500$). Each row lists the mean gap across all blocks, plus the most favorable and most unfavorable block for that variant. Discussion appears in Section 4.1.3.

Table A.1: Per-block iso-accept comparison summary on the curated stream ($M = R = 1\,500$). Mean Δ is the per-block gap averaged across all blocks. Max-block and min-block report the most favorable and most unfavorable blocks for each pair; Δ values are filter minus iso-accept random mAP.

Pair	$\overline{\Delta}$ mAP	Max block	Max Δ	Min block	Min Δ
Window single-ref	+0.0037	`arterial-rural_twi-night`	+0.008	`highway_day`	−0.002
Window two-ref	+0.0008	`city_day_clear`	+0.008	`arterial-rural_twi-night`	−0.009
Reservoir single-ref	−0.0006	`arterial-rural_twi-night`	+0.009	`highway_twi-night`	−0.010
Reservoir two-ref	+0.0031	`city_day_clear`	+0.011	`highway_twi-night`	−0.017

A.2 Streaming: per-category mAP trajectories (window pair)

Figure A.1 shows per-category mAP trajectories for window two-reference; the reservoir two-reference counterpart appears in Figure 4.5 and is discussed in Section 4.1.3.

A.3 Federated: per-category mAP trajectories (window pair)

Figure A.2 shows per-category mAP trajectories for window two-reference on the curated partition; the reservoir two-reference counterpart appears in Figure 4.13 and is discussed in Section 4.2.3.

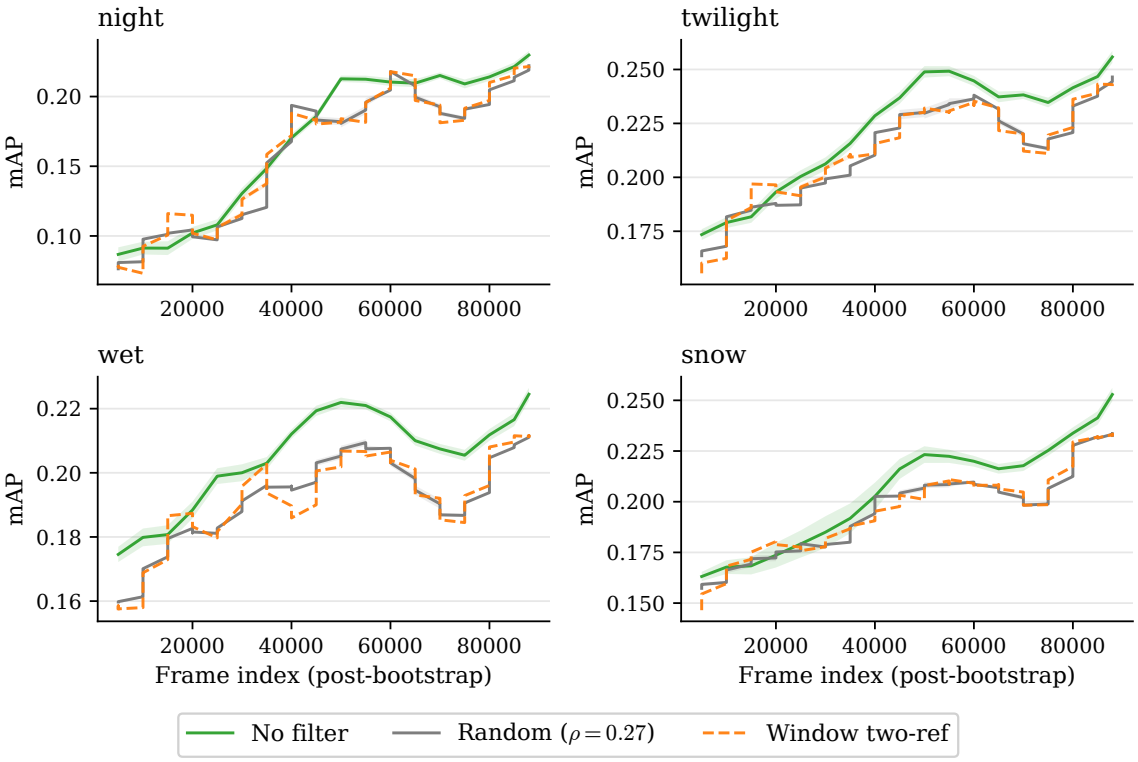

Figure A.1: Per-category mAP trajectories for window two-reference on the curated stream. Four panels show `night` and `twilight` (time-of-day buckets) and `wet` and `snow` (weather buckets). Each curve is the seed-mean mAP at every checkpoint, smoothed with a centered five-checkpoint rolling window. Faint shaded bands are the cross-seed standard deviation across three seeds. Three lines per panel: no-filter upper bound (green), window two-reference (orange dashed), and `random_p27` iso-accept random baseline at $\rho = 0.270$ (grey). Window two-reference and its random partner track each other closely on the four evaluation categories, well within the three-seed band at most checkpoints, and both stay below the no-filter upper bound by a roughly constant gap.

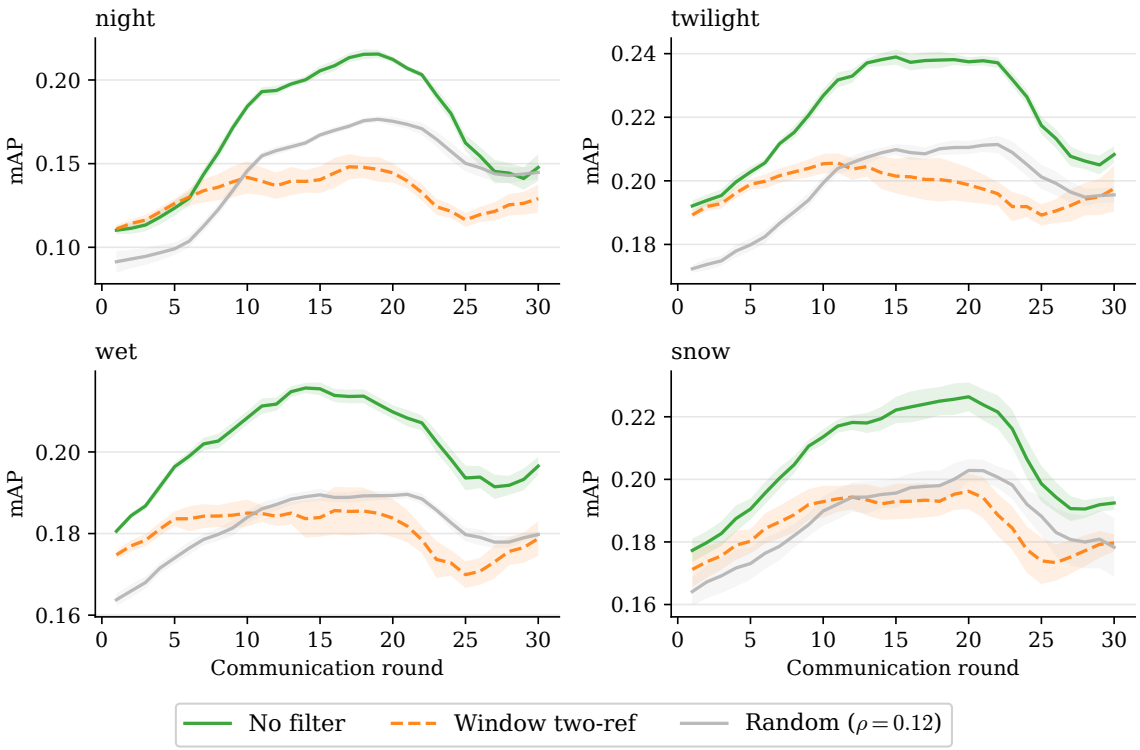

Figure A.2: Per-category mAP trajectories for window two-reference on the curated partition (default schedule), plotted in the same panels and conventions as Figure 4.13 (seed-mean lines, faint cross-seed std bands, centered five-round smoothing). Three lines per panel: no-filter upper bound (green), window two-reference (orange dashed), and `fed_random_p12` iso-accept random baseline (grey). Window two-reference opens above its random partner in the early rounds, then falls clearly below on night from round ∼10 onward, while remaining near-neutral on wet and snow.

DEPARTMENT OF ELECTRICAL ENGINEERING
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden
www.chalmers.se

CHALMERS
UNIVERSITY OF TECHNOLOGY