An Online Anti-Bunching Copilot for Bus Drivers: Streaming Headway-Risk Prediction on the SUNT Dataset

Ademar Tutor, Zaimei Xing, Subodh Kumar Dubey,
Rameesh Bandara, Praveen Vattika Venkatagiri, Phap Nguyen
Department of Computer Science, University of Waikato, Hamilton, New Zealand
COMPX523-26A: Machine Learning for Data Streams

Abstract

Bus bunching is the classic instability of high-frequency transit. A late bus collects more passengers, slows further, and is caught by the following bus. The result is long waits and uneven crowding. We propose a driver-facing *anti-bunching copilot*: an online (streaming) model that predicts, in real time, when a bus is about to bunch with the bus ahead and nudges the individual driver to ease off. The intervention is advisory and needs no new infrastructure or central control room. Using the per-vehicle *Origin–Destination* (OD) records of the Salvador Urban Network Transportation (SUNT) dataset, we reconstruct per-bus trajectories, compute the *forward headway* to the bus ahead, and frame the task as online classification of next-window bunching risk. We replay the data as a chronological stream and evaluate prequentially (test-then-train) with imbalance-aware metrics. We first show, descriptively, that the problem is real: across the eight busiest routes the within-hour headway coefficient of variation is 0.80 and irregular spacing imposes on the order of half a million avoidable passenger-waiting-minutes per day. In a bake-off of CapyMOA learners on these routes (109,029 instances), an Adaptive Random Forest (ARF) attains the best bunching-class F1 (0.916) and balanced accuracy (0.915), and — unlike the non-learning baselines — fires its warning a median of ≈28 minutes before the observed bunching, which is the property that makes a driver nudge actionable. A clearly-labelled counterfactual simulation, in which nudged drivers ease off by a few seconds, estimates a ≈13% reduction in bunching events and a ≈10% reduction in excess waiting — an effect that is a simulation under stated assumptions, not a measured field outcome.

Keywords: bus bunching; data stream mining; concept drift; ADWIN; Adaptive Random Forest; prequential evaluation; SUNT; Salvador

1 Introduction

Many people in Salvador, Brazil depend on public transport, and most of them travel by bus (Banco Nacional de Desenvolvimento Econômico e Social (BNDES) and Ministério das Cidades, 2025; Ferreira et al., 2025). The city also has heavy road traffic, and is one of the more congested cities in the world (TomTom International B.V., 2025b,a). Large transport projects are difficult to deliver here, because responsibility is split across many public and private bodies and the bus system is under serious financial strain (Tribunal de Contas do Estado da Bahia, 2022; World Bank, 2026). These conditions favour interventions that work with the buses and data that already exist, cost little, and do not rely on agencies coordinating with each other. We therefore study service reliability using the Salvador Urban Network Transportation (SUNT) dataset (Ferreira et al., 2025), and let what is realistic to deploy in this setting, not only what is technically possible, shape the design.

Within these constraints, we do not build a planner-facing dashboard, which mostly tells an authority something it would still need budget and political will to act on. Instead we target the one actor who can act *immediately* and at zero cost: the individual driver. The failure that driver can address is *bus bunching*. Headway-based services are unstable by nature: a small delay snowballs, because a late bus collects the passengers an on-time bus would have carried, which slows it further, while the bus behind finds fewer passengers, speeds up, and closes the gap. The result is one of the most common service-quality failures in urban transit. Because it can be mitigated behaviourally, by a driver easing off or briefly holding, our intervention is advisory and needs no new infrastructure.

Figure 1 contrasts the intended evenly-spaced service with the bunched state (Enayatollahi et al., 2019): in the top row, five buses are 15 minutes apart; in the bottom row, four buses have collapsed into a platoon one minute apart while the lead bus has pulled 57 minutes ahead, leaving an overcrowded front bus followed by near-empty ones. For clarity, we use the following terms throughout the report:

- **Headway:** the time gap between successive buses arriving at the same stop. Even headways are the goal of high-frequency service.
- **Bus bunching:** the unstable state in which buses that should be evenly spaced clump into a platoon, leaving a large gap behind them (Figure 1, bottom).
- **Dwell time:** the time a bus spends stopped to board and alight passengers. Heavy boarding lengthens dwell, which is a primary driver of bunching.
- **Forward headway:** the gap from a given bus to the bus *ahead* of it on the same route. This is the core signal our model predicts.
- **OD data:** the per-vehicle Origin–Destination records of SUNT, where each row is one bus arriving at one stop with its exact arrival time and onboard load.
- **Driver nudge:** the system's advisory output to the driver, namely "ease off / let the gap open" or "hold briefly at this stop". It never asks the driver to speed up.
- **Bunching-risk classes:** the three-way label the model predicts for the next window, {ok, warning, bunching}.
- **Nudge lead time:** how many minutes before an observed bunching event the warning fires. A warning is only useful if it arrives early enough for the driver to act.

This feasibility constraint also dictates the data we can use. Bunching is defined on the *headway* between two *successive* buses, which can only be measured if each bus is identified individually and its arrivals are ordered in time. The aggregated 5- and 30-minute tables provided with the assignment store only per-interval *counts*, with no individual vehicle identifier, trip identifier, or exact arrival time. As a result, when two buses pass a stop within the same window, the data cannot distinguish a 30-second gap (bunched) from a four-minute gap (normal). The aggregation

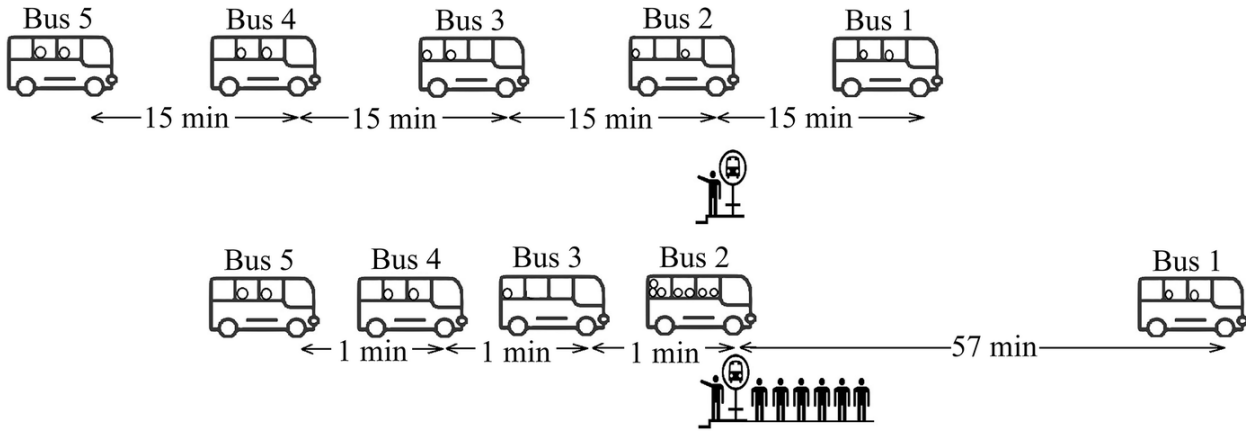

Figure 1: Evenly-spaced service (top) versus bunched service (bottom). When a bus runs late, the growing gap ahead of it lets passengers accumulate; the longer it dwells to board them, the later it gets, while the bus behind catches up. Adapted from Enayatollahi et al. (2019).

removes the very signal the task depends on. We therefore use the raw *Origin–Destination* (OD) records from the same published SUNT release, which retain per-vehicle arrivals with exact stop times and make headways directly computable; the assignment brief explicitly encourages additional public data. We set out this schema-level argument and a side-by-side comparison of the two data products in Section 3.1 (Table 1).

The problem is also a genuine *machine-learning-for-data-streams* task. Headway dynamics are non-stationary: they change with time of day, demand, traffic, weather, and network changes. A static model or fixed rule mis-fires after any shift. Online learning with *prequential* evaluation, strict chronological ordering, and drift adaptation (Bifet and Gavaldà, 2007) is the appropriate tool, and the per-bus stream is naturally replayed from SUNT by timestamp. A small in-cab prototype could surface the model's output as a glanceable risk light and a short nudge. In what follows we (i) establish descriptively that bunching is a real and costly problem on these routes, (ii) bake off CapyMOA learners under prequential evaluation to find one that warns early enough to act on, and (iii) estimate the plausible effect of a small, bounded driver ease-off in a clearly-labelled counterfactual simulation.

2 Related Work

This work sits at the intersection of transit operations and machine learning for data streams, and we position it along three lines. We first place bunching in its wider operational context, namely how bus operations and bunching interact with road traffic (Section 2.1). We then review how prior work predicts and controls bunching, and where those approaches assume infrastructure or central authority that the Salvador setting lacks (Section 2.2). Finally we turn to the data-stream learning methods this study adapts to the problem (Section 2.3). Across all three, the gap we address is the same: prior work tends to act centrally or offline, whereas we target a decentralised, online, per-driver nudge.

2.1 Buses, bunching, and road traffic

Bus operations interact with the surrounding road traffic. This matters for an intervention that adjusts a bus's pace. Nguyen-Phuoc et al. (2018) estimate the *net* effect of bus operations on network congestion and find it positive overall. In Melbourne, buses cut severely congested links by roughly a tenth by keeping would-be drivers off the road. At the same time, the congestion *cost* of buses is concentrated at stops, where stop-start dwell reduces road capacity. This is relevant here because dwell is also the mechanism that drives bunching. The same stop dynamics therefore

underlie both phenomena, and improving bus reliability may help protect the road-congestion benefit that buses provide. A natural concern is that an intervention which deliberately delays a bus could worsen road traffic. Qiang and Huang (2023) address this concern with a cellular-automaton simulation of *mixed* traffic. They show that self-equalising headway control continues to regulate bus spacing on non-dedicated lanes. It also does not appear to harm car speed or overall capacity, and at intermediate density it may modestly improve them, though this evidence comes from an idealised synthetic route rather than field data. Both studies treat the bus and traffic interaction with offline planning or simulation models and with network- or fleet-level levers. This is useful for bounding the wider congestion effects of bus operation. It leaves open the per-vehicle, real-time setting that this study targets, in which the lever is a single driver's voluntary ease-off rather than a centrally imposed policy.

2.2 Predicting and controlling bus bunching

Bunching and headway instability are long studied. Enayatollahi et al. (2019) characterise the headway as intrinsically unstable and show that even a small reduction in per-passenger boarding time, or limiting a delayed bus's dwell at stops, can markedly reduce bunching. Classical *control* schemes act by delaying a bus rather than speeding one up: Daganzo (2009) proposed adaptive holding/headway control that keeps spacing even, Bartholdi III and Eisenstein (2012) make headways *self-equalising* by holding each arriving bus in proportion to the headway behind it, and Daganzo and Pilachowski (2011) achieve the same end through two-way speed cooperation, in which a leading bus eases off for the one behind but is never asked to exceed its normal speed. This "slow the leader, never the follower" principle is the action our advisory approximates. A separate *prediction* line forecasts bunching before it forms: Yu et al. (2016) predict stop-level headway from smart-card data and flag over 95% of events several stops ahead, and Borges Santos et al. (2022) predict bunching up to ten stops ahead with a tree ensemble on Brazilian GPS/GTFS data. These predict but do not act, and Moreira-Matias et al. (2016) close the loop with an online learning framework that detects and counters bunching in real time, reporting large reductions in bunching and meaningful waiting-time savings. Most such schemes nonetheless assume a *central controller* issuing holds; our contribution is a driver-personal, decentralised nudge driven by an online stream model, with no control room.

2.3 Adapting machine learning for data streams to bus bunching

Predicting bunching online means treating each bus arrival as one event in an unbounded, non-stationary stream, which calls for learners that update incrementally and adapt to change. Hoeffding Trees learn decision trees incrementally from unbounded streams using the Hoeffding bound to decide splits (Domingos and Hulten, 2000). The adaptive variant couples the tree with ADWIN-monitored branches that self-replace under drift (Bifet and Gavaldà, 2009). Adaptive Random Forest (ARF) extends streaming forests with online bagging, random feature subspaces, per-tree drift detection, and weighted voting (Gomes et al., 2017). Because our positive class is rare and its prevalence drifts across the day, the relevant subfield is learning from *imbalanced* data streams, which is now systematically benchmarked: Aguiar et al. (2024) compare 24 stream classifiers across 515 imbalanced, drifting benchmarks and recommend reporting complementary, imbalance-aware metrics rather than accuracy, while Wang et al. (2015) give the canonical online-resampling baselines (OOB/UOB) that re-tune sampling as the imbalance ratio shifts. We access these learners through CapyMOA (CapyMOA Developers, 2024), which exposes the MOA engine in Python with prequential tooling. Underpinning these adaptive learners, ADWIN detects distribution change with an adaptive window and guarantees (Bifet and Gavaldà, 2007); we use it both inside the learners and as a drift log, so the model tracks the time-of-day and demand shifts that bunching dynamics exhibit.

Table 1: What bunching detection needs, and whether each data product provides it.

Required signal	Raw OD	5-/30-min aggregation
Individual vehicle id	yes	no (count only)
Individual trip id	yes	no (count only)
Exact arrival time	yes	no (interval label)
Order successive buses	yes	no
Compute headway	**yes**	**impossible**

3 Methodology

Our central idea is to avert bunching before it forms, rather than reacting to it after the fact: because a bunching event begins as a single bus drifting off its headway, it can be averted by a small, timely adjustment from that one driver. Realising this idea has four requirements, and this section addresses them in turn. We first establish what data the approach needs and why the chosen source supplies it (Section 3.1), since the whole method is built on a quantity that the data must let us measure. We then present the streaming driver-nudge system itself (Section 3.2), the preprocessing that turns raw operational records into the stream the model consumes (Section 3.4), and the counterfactual simulation used to estimate the effect of acting on its predictions (Section 3.6).

3.1 Data requirements and source

The approach hinges on the *headway*: the time gap between two *successive* buses arriving at the same stop, which is the quantity bunching is defined on. Before describing the system, we therefore establish what data is needed to measure the headway, and why the source we use supplies it while the data provided by the assignment does not.

To measure the headway, the data must let us identify each bus individually and order its arrivals in time. Concretely, every record needs the individual vehicle, the individual trip, and the exact arrival time at the stop. Without all three, successive buses cannot be told apart and the gap between them cannot be computed (Table 1).

The assignment supplies a 10-stop subset of SUNT in two pre-aggregated versions, at 5-minute and 30-minute resolution. These tables are keyed by (stop, time-interval) and store only per-interval *counts*, such as boardings, loading, and the number of vehicles. They carry no individual vehicle identifier and no exact arrival timestamp. If two buses pass a stop within the same 5-minute window, the aggregated data cannot tell whether they were 30 seconds apart (bunched) or 4 minutes apart (normal). The aggregation therefore destroys the very signal bunching is defined on, and no amount of post-processing can recover it.

We therefore use the raw *Origin–Destination* (OD) table from the same published SUNT release (Ferreira et al., 2025). This is the source from which the provided subset was derived, and the assignment brief explicitly encourages using additional public data, so the OD table is squarely in scope. Each OD row is one bus arriving at one stop, carrying the vehicle, the trip identifier, the exact `stop_time`, and the onboard `loading`. These are exactly the fields the aggregated subset lacks, so successive-bus headways become directly computable. To keep the experiment tractable we scope it to the eight busiest routes, both directions, over 1–3 March 2024, rather than the whole network.

We use the Origin–Destination (OD) tables of the SUNT dataset. In this repository the raw files are at

`Dataset/SUNT/data/od/od-YYYY-MM-DD.parquet`

one Apache Parquet file per calendar day, 183 files in total spanning 2024-03-01 to 2025-03-14 (≈1.4 GB). The experiment configuration (`experiments/anti-bunching-copilot/config.yaml`,

key `paths.od_dir`) points at this folder. We scope the study to the eight busiest routes (1007, 1386, 1346, 1649, 12503, 1320, 1637, 1413), both directions, over 1–3 March 2024.

Each row of an OD file is *one bus arriving at one stop*: which bus, on which route and trip, at which stop, at exactly what time, and how many people boarded, alighted, or were on board. From the arrival times of successive buses at the same stop we reconstruct the *headway*, the gap to the bus in front, which is the quantity bunching is defined on. Table 2 lists every raw column, and Table 3 shows eleven real consecutive rows so the reader can see what the data actually looks like.

Table 2: Raw OD schema (`od-YYYY-MM-DD.parquet`). Boarding/alighting columns are hyphenated.

Column	Type	Meaning (plain English)
`route_short_name`	string	The bus line, e.g. "1007".
`register_code`	int	Internal service-registration code for the run.
`direction_id`	string	Direction of travel: "I" = outbound, "V" = return.
`pt_sequence`	int	The stop's position along the trip (1 = first stop).
`stop_id`	int	Which physical stop this is.
`vehicle`	int	Which bus (vehicle) this is, which lets us follow one bus over time.
`trip_number`	int	The vehicle's n-th trip of the day.
`trip_id`	string	One run of one bus, coded `{vehicle}_{route}_{seq}`.
`start_trip`	datetime	When this trip began (leaves the first terminal).
`end_trip`	datetime	When this trip ended (reaches the last terminal).
`stop_time`	datetime	The exact time this bus arrived at this stop, the key signal.
`n-boardings`	float	How many passengers got on at this stop.
`n-alighting`	float	How many passengers got off at this stop.
`lag_loading`	float	How full the bus was *just before* it arrived.
`balance`	float	Load after people got off (= `lag_loading` − `n-alighting`).
`loading`	float	Load after people got on (= `balance` + `n-boardings`); i.e. how full it leaves.

Table 3: Eleven real consecutive OD rows, the first stops of one trip (vehicle 20407, route 1007, direction I, on 2024-03-01). Each row is one stop arrival. Columns are abbreviated: `reg` = register_code, `seq` = pt_sequence, `veh` = vehicle, `trip#` = trip_number, `board/alight` = n-boardings/n-alighting, `lag/bal/load` = lag_loading/balance/loading. The `start`, `end` and `time` columns are times on 2024-03-01. Watch the headway emerge as `time` advances down the rows, and the load build as people board. The shaded row (`seq` 8) is the one traced in the worked example.

route	dir	reg	seq	stop_id	veh	trip#	trip_id	start	end	time	board	alight	lag	bal	load
1007	I	73410	1	230377744	20407	2	20407_1007_2	13:22:03	14:48:51	13:22:03	0.0	0.0	0.0	0.0	0.0
1007	I	73410	2	193563134	20407	2	20407_1007_2	13:22:03	14:48:51	13:22:23	0.0	0.0	0.0	0.0	0.0
1007	I	73410	3	304938514	20407	2	20407_1007_2	13:22:03	14:48:51	13:25:43	0.0	0.0	0.0	0.0	0.0
1007	I	73410	4	304938515	20407	2	20407_1007_2	13:22:03	14:48:51	13:26:53	0.0	0.0	0.0	0.0	0.0
1007	I	73410	5	304938513	20407	2	20407_1007_2	13:22:03	14:48:51	13:27:53	0.0	0.0	0.0	0.0	0.0
1007	I	73410	6	202057097	20407	2	20407_1007_2	13:22:03	14:48:51	13:29:04	0.0	0.0	0.0	0.0	0.0
1007	I	73410	7	277469508	20407	2	20407_1007_2	13:22:03	14:48:51	13:29:25	0.0	0.0	0.0	0.0	0.0
1007	I	73410	8	230377745	20407	2	20407_1007_2	13:22:03	14:48:51	13:30:25	2.3	0.0	0.0	0.0	2.3
1007	I	73410	9	304939642	20407	2	20407_1007_2	13:22:03	14:48:51	13:32:17	0.0	0.0	2.3	2.3	2.3
1007	I	73410	10	193563135	20407	2	20407_1007_2	13:22:03	14:48:51	13:32:38	0.0	0.0	2.3	2.3	2.3
1007	I	73410	11	230377743	20407	2	20407_1007_2	13:22:03	14:48:51	13:33:26	1.1	0.0	2.3	2.3	3.4

3.2 A streaming driver-nudge system

To address bus bunching under the constraints set out in Section 1, where the only readily available corrective action is a driver easing off or briefly holding and no new infrastructure can be assumed, we propose a system that catches bunching as it begins and prompts that small correction in time. The system is a real-time *warning and suggestion* aid for the driver. As a bus travels its route, the system continuously estimates its risk of bunching over the next few stops and, when that risk is high, issues a short suggestion such as "ease off" or "hold briefly". It is advisory only: it informs the driver, who remains in control. The same predictions could equally feed a control-room view, but we deliberately target the driver because that is where a zero-cost corrective action is available, and because it does not depend on the cross-agency coordination that the Salvador setting makes difficult.

Bus operation is an unbounded, non-stationary stream of events: each bus arrival produces a new observation, and the relationships that signal impending bunching shift with time of day, demand, traffic, and weather. A model trained once on a fixed sample would grow stale. We therefore use *online* (data-stream) machine learning, which updates with every new arrival and adapts to drift, and we evaluate it *prequentially* (test-then-train), so the reported performance is the performance the system would have achieved had it been running live. This matches the course's focus on learning from evolving data streams and is the natural fit for a tool meant to run continuously on board.

3.3 Driver nudging as online classification

We cast the problem as online classification. For each active bus, at each stop arrival, the model predicts the *next-window bunching risk* in three classes {ok, warning, bunching}: will the bus's forward headway fall below a fraction of the rolling local headway within the next few stops? A `warning` or `bunching` prediction triggers a driver nudge.

The nudge translates a risk prediction into a concrete, safe action. When the bus is *closing* on the bus ahead, the system suggests "ease off / let the gap open"; when a gap is opening behind it (running hot), it suggests "hold briefly at this stop". The action space is deliberately safe: ease off or briefly hold, never speed up. In the in-cab prototype this surfaces as a glanceable risk light and a one-line message, delivered by a small web dashboard that replays the stream and visualises each bus's predicted risk in near real time. Figure 2 shows the runtime flow end to end, from the data the model ingests to the warning the driver receives.

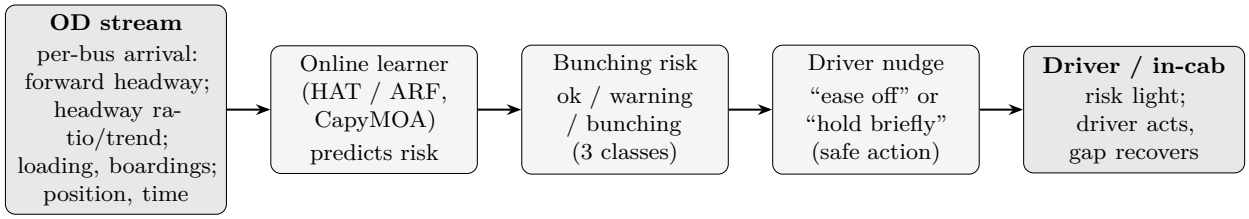

Figure 2: Runtime flow of the driver-nudge system. The online learner ingests the per-bus features of each arrival, predicts the next-window bunching risk, and a `warning` or `bunching` class triggers a safe nudge that the driver can act on so the impending bunching is averted.

Table 4 makes the chain concrete: as a bus drifts toward the bus ahead, its headway ratio falls, the model's prediction escalates from `ok` to `warning` and then `bunching`, and a nudge fires. A warning is only useful, though, if it arrives early enough to act on. We therefore care about the *nudge lead time*: the gap between the first fired warning and the first observed bunching on a trip (Figure 3). A larger lead time gives the driver more room to ease off before the buses close up, and it is one of the metrics we report in Section 4.

Table 4: Illustrative worked example (not measured) of consecutive arrivals on one trip. As the headway ratio falls toward the bunching threshold, the model's prediction flips to `warning` and then `bunching`, and a nudge fires. The lead time is the gap from the first fired warning to the first observed bunching.

Stop (`pt_seq`)	Time (min)	Headway (min)	Headway ratio	True label	Model prediction	Nudge fired?
12	0	9.8	0.98	ok	ok	—
13	6	7.1	0.71	warning	warning	yes (ease off)
14	11	5.4	0.54	warning	warning	yes (ease off)
15	17	3.6	0.36	bunching	bunching	yes (ease off)
16	23	2.9	0.29	bunching	bunching	yes (ease off)

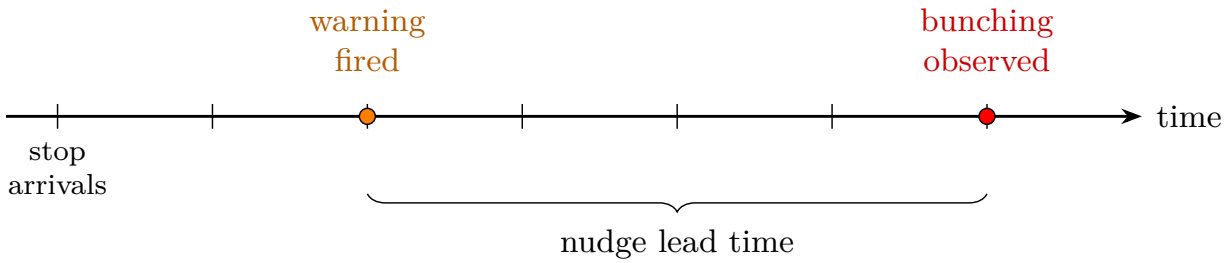

Figure 3: How nudge lead time is measured (illustrative). Along one trip, it is the elapsed time from the earliest fired warning to the first observed bunching event. A larger lead time gives the driver more opportunity to ease off before bunching forms.

3.4 Preprocessing

Target format and rationale The model does not consume the raw OD rows directly. Preprocessing transforms them into a tidy *labelled instance stream*: one row per bus arrival at a stop, each carrying a fixed numeric feature vector and a class label, with all rows sorted by `stop_time` into true chronological order. This format is dictated by the streaming setting. First, prequential (test-then-train) evaluation requires the data to be a sequence of $(\mathbf{x}_t, y_t)$ events that can be replayed one at a time in the order they occurred; a row-per-arrival, time-ordered table is exactly that. Second, chronological order is what prevents leakage: the learner only ever sees the past before predicting the next arrival, mirroring deployment. Third, the quantity of interest, the headway, is not a column in OD and must be *derived* (Section 3.1), so a construction step is unavoidable. Algorithm 1 gives the procedure, Figure 4 the high-level flow, and Table 5 the resulting fields.

Algorithm 1: Constructing the streaming dataset from SUNT OD.

Input: OD event rows; scope (routes, directions, dates); headway window w; look-ahead K stops; thresholds $\alpha < \beta$ (we use α=0.40, β=0.60); minimum plausible gap g

Output: `features.csv`: one labelled instance per bus arrival, sorted by `stop_time`

Load the OD parquet files for the dates; keep the scoped routes and directions; parse `stop_time`

foreach *trip* **do**
  if *`stop_time` is not non-decreasing along the stop sequence* **then** discard the trip

foreach *trip* **do**
  compute the segment time to the next stop and the time since the previous stop

foreach *(route, direction, stop), in time order* **do**
  $h \leftarrow$ arrival time − previous distinct vehicle's arrival time `// forward headway`
  if $h < g$ **then** $h \leftarrow$ undefined `// same-vehicle artefact`
  $m \leftarrow$ median of the previous w headways at this stop `// local "normal"`

foreach *arrival on a trip* **do**
  $h_{\min} \leftarrow$ minimum forward headway over the next K stops `// future risk,` $> t$
  if $h_{\min} < \alpha\, m$ **then**
    $y \leftarrow$ *bunching*
  else if $h_{\min} < \beta\, m$ **then**
    $y \leftarrow$ *warning*
  else $y \leftarrow$ *ok*

Engineer the causal features of Table 5 (all $\leq t$): headway ratio, trend
Drop arrivals with no defined headway, normal, or label; replace any non-finite values
Sort the surviving rows by `stop_time`

3.5 How the algorithm builds the stream

In plain terms, Algorithm 1 walks the OD events once and turns them into a clean, time-ordered table of labelled examples. It first discards trips whose timestamps are out of order, since any headway computed from them would be wrong. For every stop it then measures the *forward headway*, the gap between each bus and the one before it, and a rolling "normal" headway for that stop (the median of recent gaps) as a local baseline. Each arrival is then labelled by looking a few stops ahead: if the bus is about to fall well below its normal spacing it is marked *bunching*, if it is moderately below it is marked *warning*, and otherwise *ok*. Crucially the label looks into the future while the features only use the past, so there is no leakage. Finally the algorithm attaches the causal features, drops arrivals that cannot be labelled (such as the first bus of the day at a stop), and sorts everything by time.

To make this concrete, we trace the same eleven raw rows of Table 3 through preprocessing. Table 5 lists the columns we engineer from the raw rows into the modelling table `data/processed/features.csv` (109,029 rows for this scope), and Table 6 shows those *same* eleven rows after preprocessing, so the reader can trace raw data straight into the features the model consumes. Boarding and loading counts are fractional because SUNT *infers* them from fare and vehicle data, so a value like `2.3` is expected, not a typo.

As a worked example, follow stop 8 (`seq` 8) through both tables. In the raw row (Table 3) the bus arrives at `13:30:25` carrying `load` 2.3. Preprocessing turns that into the matching row of Table 6: the arrival time becomes `min_day` = 810 (since $13 \times 60 + 30 = 810$); the gap to the previous bus at this same stop is `headway` = 15.8 min, while the local "normal" there is `normal` = 39.4 min, so `ratio` = 15.8/39.4 ≈ 0.40, meaning the bus is at about 40% of its usual spacing, i.e. closing on the bus ahead; the time since the previous stop is `since_prev` = 1.0 min (from `13:29:25` to `13:30:25`); the onboard load carries over as `load` = 2.26 (the same 2.3, at full precision); the date is a Friday so `dow` = 4, and 13:30 is outside the commute peaks so `peak` = 0. Finally the `label` is 1 (warning): looking a few stops ahead, the headway is about to fall further toward the bunching threshold. In short, one raw OD arrival becomes one row of model-ready numbers.

Table 5: Engineered columns in `features.csv`. "Causal" columns use only the past ($\leq$ now); the `label` looks into the future and is the prediction target, never an input.

Column	Role	Meaning (plain English)
`headway_min`	feature	Minutes since the *previous* bus passed this same stop, the gap to the bus ahead.
`local_median_headway`	reference	The "normal" gap here: the median of the last 8 gaps at this stop.
`headway_ratio`	feature	`headway_min` ÷ normal. Below 1 = closer than usual (bunching).
`headway_trend`	feature	Change in the gap versus the previous bus at this stop (closing or opening).
`since_prev_stop_min`	feature	Minutes this bus took to get here from its previous stop (running-time proxy).
`pt_sequence`	feature	Position along the route.
`is_peak`	feature	1 during the commute peaks (06–08h, 17–19h), else 0.
`label`	target	{0 ok, 1 warning, 2 bunching}: is the gap about to collapse below 60%/40% of normal within the next 3 stops?

Table 6: The engineered `features.csv` rows computed from the *same* eleven raw OD rows of Table 3 (trip 20407_1007_2). Read the two tables together: Table 3 is the raw input, this is what the model sees after preprocessing (Table 5 defines each column). Abbreviations: `seq` = pt_sequence, `headway` = headway_min, `normal` = local_median_headway, `ratio` = headway_ratio, `trend` = headway_trend, `since_prev` = since_prev_stop_min, `peak` = is_peak. Note how the raw `stop_time` of Table 3 turns into the per-stop gaps and into `since_prev`; and a bunched `ratio` ($\approx$0.4) yields warning (1) or bunching (2) labels. The shaded row (`seq` 8) is the one traced in the worked example.

seq	headway	normal	ratio	trend	since_prev	min_day	peak	label
1	18.07	37.00	0.49	-11.88	0.00	4	0	1
2	18.13	37.14	0.49	-12.08	0.33	4	0	1
3	16.87	38.92	0.43	-17.93	3.33	4	0	1
4	16.75	39.02	0.43	-19.32	1.17	4	0	1
5	16.25	39.36	0.41	-21.30	1.00	4	0	1
6	16.22	40.14	0.40	-22.53	1.18	4	0	2
7	15.75	40.48	0.39	-23.80	0.35	4	0	2
8	15.82	39.43	0.40	-22.55	1.00	4	0	1
9	16.35	38.34	0.43	-21.42	1.87	4	0	1
10	16.33	38.38	0.43	-21.47	0.35	4	0	1
11	16.35	34.40	0.48	-10.13	0.80	4	0	1

The output is `features.csv`: one row per bus arrival, in the exact order the arrivals happened. This file *is* the stream that the learners consume in Section 4. They read it one row at a time and, for each arrival, first predict the bunching class from the features and then learn from the true label, mirroring how the system would run live (prequential, test-then-train). Because the table is already ordered and labelled, no further reshaping is needed: the preprocessing is what turns raw operational data into something a streaming model can learn from online.

3.6 Counterfactual simulation design

Predicting bunching is only useful if acting on the prediction would change the outcome, yet the true effect of drivers easing off cannot be observed from historical data: the public SUNT release contains no period in which the copilot was deployed. We therefore design a *counterfactual*

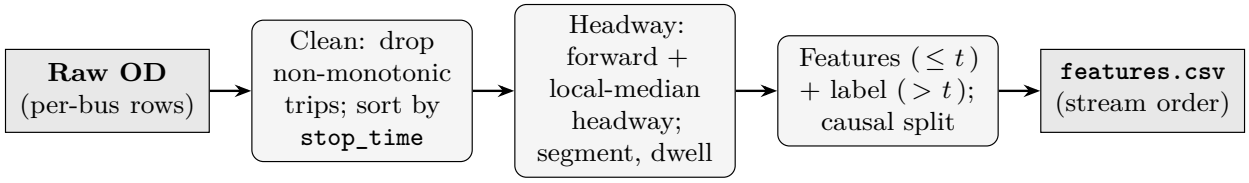

Figure 4: Preprocessing pipeline from the raw SUNT OD data to the modelling table. Each stage is a function in the submitted code; the output is replayed chronologically for prequential evaluation.

simulation that re-runs the actual headway geometry under a small, bounded behavioural change, rather than asserting an effect. This subsection specifies the procedure; its results are reported in Section 5.2 and interpreted in Section 6.

The simulation replays the recorded arrival stream in time order. Whenever the deployed ARF fires a nudge on a bus that is genuinely *closing* on the one ahead (headway ratio below the warning threshold), the driver eases off: we add a small bounded delay — a few seconds per stop, capped at 1.5 min per trip — to that bus's subsequent arrivals, then recompute the downstream forward headways and re-measure bunching, headway CV, and excess wait (Eq. 2). The ease-off magnitude is the decentralised analogue of the small adaptive holds of Daganzo (2009) and the advisory counterpart of the cooperative speed control of Daganzo and Pilachowski (2011), in which a leading bus slows for the one behind and is never asked to speed up. It is also consistent with Enayatollahi et al. (2019), who show that shaving a little dwell per stop materially reduces bunching. The nudge never asks a driver to speed up, and the local "normal" headway is held at its baseline value so the comparison is against a fixed reference. Algorithm 2 states the procedure.

Algorithm 2: Counterfactual ease-off replay (simulation).

Input: recorded arrivals with model predictions; ease-off δ (s); per-trip cap C; closing threshold τ; mode $\in$ {all-fired, true-positive-only}
Output: counterfactual bunching count, headway CV, and excess wait vs. baseline

$delay[trip] \leftarrow 0$ for all trips
foreach *arrival a in trip order* **do**
  $t'[a] \leftarrow$ `stop_time`$[a] + delay[trip(a)]$ `// shift by accumulated ease-off`
  $fired \leftarrow$ (prediction$[a] \geq$ warning)
  if *mode = true-positive-only* **then** $fired \leftarrow fired \wedge$ (label$[a] \geq$ warning)
  $closing \leftarrow$ (headway ratio$[a] < \tau$)
  if $fired \wedge closing \wedge delay[trip(a)] < C$ **then**
    $delay[trip(a)] \leftarrow \min(delay[trip(a)] + \delta,\ C)$

recompute forward headways on the shifted times t'; recompute bunching, CV, excess wait

To keep the estimate honest we build in two robustness checks, reported in Section 5.2: an *ablation* that gates the ease-off to only the model's *correct* (true-positive) predictions, isolating the contribution of real-time prediction from hindsight; and a *sensitivity sweep* of the per-stop ease-off magnitude, testing whether the result depends on one chosen value. Two effects are deliberately outside the simulation's reach: real driver compliance and the true human response are not in historical data, so the result is an estimate under stated assumptions rather than a measured field outcome; and bus occupancy and stop boardings have no measurable link to spacing in SUNT ($r \approx 0.09$ and $r \approx 0.00$), so we do not simulate a crowding benefit.

4 Experiments

Setup We reconstruct per-bus trajectories from OD, compute forward headways per route×direction×stop, build the causal features and the forward-looking label, and replay all events chronologically by `stop_time`. The reported run scopes to the eight busiest routes on 1 March 2024 — 1007, 1386, 1346, 1649, 12503, 1320, 1637 and 1413 — over 1–3 March 2024 (109,029 instances; class balance: ok 70%, warning 10%, bunching 20%). Trips with non-monotonic `stop_time` (≈23% at this scope)

Table 7: Prequential model bake-off (bunching class), 109,029 instances. Ranked by F1. The static threshold is a non-learning reference. Throughput (inst./s) is wall-clock and hardware-dependent, so the absolute values are indicative rather than reproducible; only the relative ordering (e.g. HAT roughly an order of magnitude faster than ARF) is meaningful.

Model	F1	Recall	Bal. acc.	κ	inst./s
ARF	**0.916**	0.915	0.915	0.892	6,162
static threshold (ref.)	0.909	0.890	0.904	0.883	—
HAT	0.907	0.901	0.899	0.879	57,658
Hoeffding Tree	0.886	0.848	0.791	0.792	71,790
EFDT	0.877	0.836	0.789	0.785	63,120
KNN	0.761	0.721	0.728	0.676	11,575
Naive Bayes	0.000	0.000	0.333	0.000	97,835
Majority class	0.000	0.000	0.334	0.002	86,598

are dropped. Before modelling we establish, purely descriptively, that bunching is a real and costly problem on these routes (Section 5.1).

Evaluation We use *prequential* (test-then-train) evaluation in strict chronological order: the model predicts for the bus about to arrive, the bus arrives, the model learns from the realised outcome, then predicts the next. Random train/test splits would leak the future and are avoided. Because bunching is the rare positive class, we report imbalance-aware metrics — precision, recall, and F1 on the bunching class, plus balanced accuracy and Cohen's κ — following established practice for evaluating imbalanced data streams (Aguiar et al., 2024), where accuracy alone is misleading under a drifting class ratio. We also report the *nudge lead time*: the minutes between a fired warning and the observed bunching (introduced in Section 3.2, Figure 3).

Baselines and bake-off The streaming models are compared against deliberately simple alternatives: a static headway-threshold rule ("is the gap small right now?"), previous-headway persistence, and a majority-class floor. We then run a bake-off of CapyMOA classifiers on the same stream: Naive Bayes, k-Nearest Neighbours, Hoeffding Tree, Extremely Fast Decision Tree (EFDT), Hoeffding Adaptive Tree (HAT), and Adaptive Random Forest (ARF).

Hyperparameters ARF uses an ensemble size of 25 and Poisson $\lambda = 6$; trees use a grace period of 200; ADWIN uses $\delta = 0.002$. A fixed random seed is reported for reproducibility.

4.1 Bake-off results

We first report which learner predicts bunching best, and early enough to act on, before turning to the service-quality measurements in Section 5. Table 7 ranks the candidates by bunching-class F1. ARF attains the best F1 (0.916), balanced accuracy (0.915), and κ (0.892), ahead of the single adaptive tree (HAT) and the static threshold. The non-learning Naive Bayes and majority-class floor collapse to the majority class and score zero on the bunching class. There is a clear accuracy-versus-throughput trade-off: HAT processes roughly an order of magnitude more instances per second than ARF for about one F1 point less, which matters for an in-cab deployment.

Crucially, the streaming models warn *ahead of time*. ARF's nudges fire a median of ≈28 minutes before the observed bunching (Fig. 6), whereas the static threshold and persistence baselines have no meaningful lead — they detect bunching as it happens rather than predicting it, so they cannot support a useful driver nudge. Figure 5 shows prequential accuracy over the stream with ADWIN drift markers, confirming the adaptive learners recover after distribution shifts.

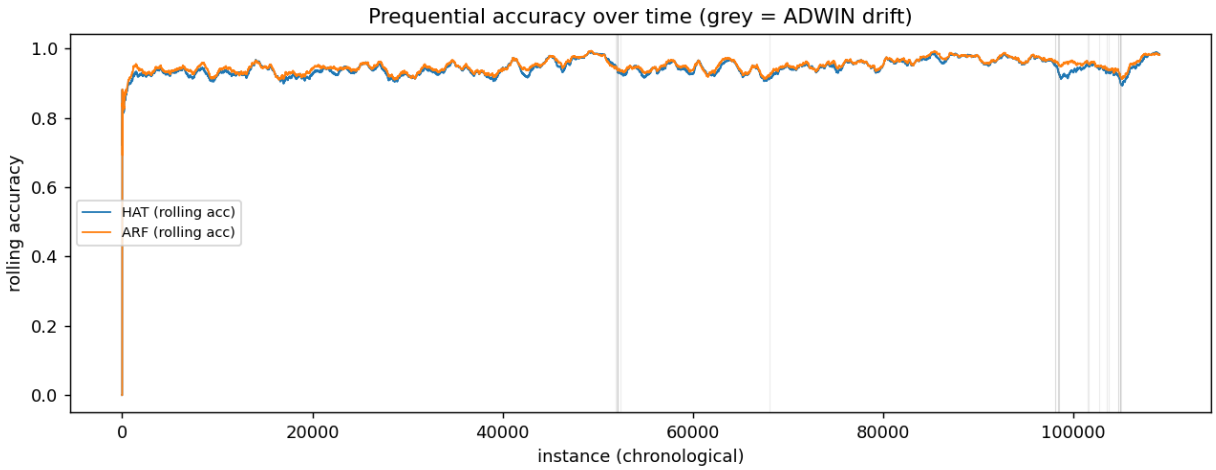

Figure 5: Prequential accuracy over the chronological stream for HAT and ARF; grey lines mark ADWIN-detected drift points.

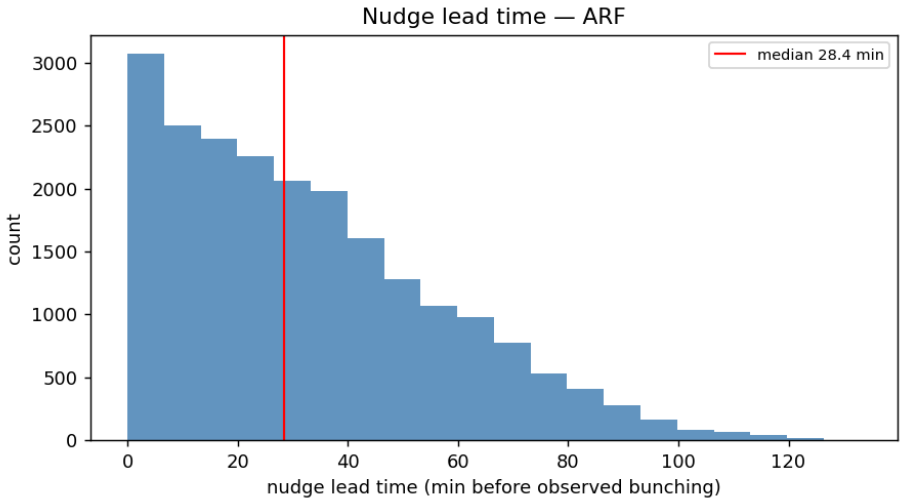

Figure 6: Nudge lead time for ARF: minutes between a fired warning and the observed bunching.

5 Results

Having selected a model that predicts bunching with actionable lead time (Section 4.1), we now measure the service-quality effect. We present this as a *before/after* comparison: Section 5.1 measures how bunched the recorded service already is (the "before"), and Section 5.2 re-runs each of those measurements on the counterfactual "after" (Section 3.6), stating for each whether it improved. Section 5.2 mirrors Section 5.1 measurement-for-measurement, and Table 9 maps the pairs and their verdicts at a glance.

5.1 Before: bunching in the recorded data

We first establish, *descriptively and without machine learning*, that bunching is a real and costly problem on these routes, measuring it directly from the reconstructed forward headways. To avoid mistaking the natural daily demand cycle (short peak gaps, long late-evening gaps) for bunching, every statistic is computed *within* hourly windows — per route, direction, stop and hour-of-day — so we compare like-with-like demand. We report six measurements; the same six are re-run after the nudge in Section 5.2.

5.1.1 Headway irregularity (CV)

The standard measure of spacing irregularity is the headway coefficient of variation CV = std(h)/mean(h): CV = 0 is perfectly even service and bunching inflates it. Across the eight routes the median within-window CV is 0.73 (mean 0.80); the irregularity is worse in the peak, where most passengers travel, rising from 0.79 off-peak to 0.86 in the peak (Fig. 8).

5.1.2 How close the buses run (headway ratio)

For every arrival we compute the headway ratio (realised gap divided by the local-normal gap). By the same thresholds the model uses, 19.4% of arrivals are already bunched (ratio < 0.40) and a further 9.8% are in warning (ratio < 0.60); the full distribution is Fig. 7.

5.1.3 Avoidable passenger waiting

We convert this irregularity into a concrete cost — minutes of avoidable waiting. For a stop served at headways h_i, the mean wait of a passenger arriving at random (Poisson) is the renewal-theory result

$$\mathbb{E}[W] \;=\; \frac{\mathbb{E}[h^2]}{2\,\mathbb{E}[h]} \;=\; \frac{\bar{h}}{2}(1+\mathrm{CV}^2), \tag{1}$$

whose first term $\bar{h}/2$ is the wait under perfectly even service. The remainder is therefore the *avoidable* wait caused purely by irregularity,

$$W_{\text{excess}} \;=\; \frac{\bar{h}}{2}\,\mathrm{CV}^2. \tag{2}$$

Applying Eq. (2) per window and weighting by boardings, irregular spacing on these eight routes imposes on the order of 5.6×10^5 avoidable passenger-waiting-minutes per day ($\approx$ 9,400 passenger-hours daily), with a mean of ≈ 17 avoidable minutes per waiting passenger (Fig. 9).

5.1.4 Severity over the day

Following Enayatollahi et al. (2019), we track a per-arrival severity max(0, 1 − headway ratio), which is 0 on schedule and rises toward 1 as a bus closes up on the one ahead. Averaged by hour it peaks in the morning and evening commute (Fig. 10), so bunching concentrates exactly when buses are fullest.

5.1.5 Seeing bunching directly (string diagram)

The per-bus string diagram (Fig. 11) shows the same phenomenon in the raw trajectories: time runs along the horizontal axis and a bus's position along the route up the vertical axis, so each line follows one bus, and where lines converge or cross those buses are bunching. This structure is visible only because OD records each bus individually with exact arrival times, and would vanish under 5-/30-minute aggregation (Section 3.1).

5.1.6 Travel-time context (congestion, not a nudge target)

For completeness we also record terminal-to-terminal ride time (`end_trip` − `start_trip`), which is longer at peak on most routes (Table 8): route 1007 is +18.4%, the network median +2.7%. This is ordinary road congestion. We include it as context only: the nudge evens *spacing*, not driving speed, so this is explicitly *not* a quantity we expect it to improve — a point we make good on in Section 5.2.6.

Table 8: Travel-time context (“before”): median terminal-to-terminal trip time, peak versus off-peak. Real congestion, but not a target of the nudge (cf. Section 5.2.6).

Route	Peak (min)	Off-peak (min)	Δ
1007	99.5	84.1	+18.4%
1346	49.5	45.0	+9.9%
1413	88.1	82.8	+6.4%
1320	66.3	65.0	+2.0%
12503	95.0	93.4	+1.7%
1386	92.6	91.1	+1.6%
1649	83.3	84.1	−0.9%
1637	78.1	79.1	−1.3%

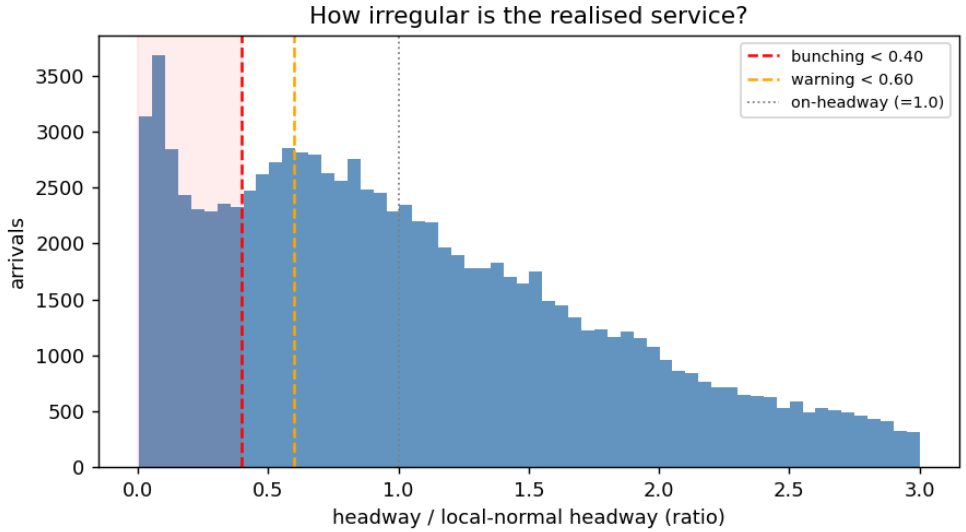

Figure 7: Distribution of the realised headway as a fraction of the local-normal headway. Mass left of the dashed lines is service that is in warning (< 0.60) or bunched (< 0.40).

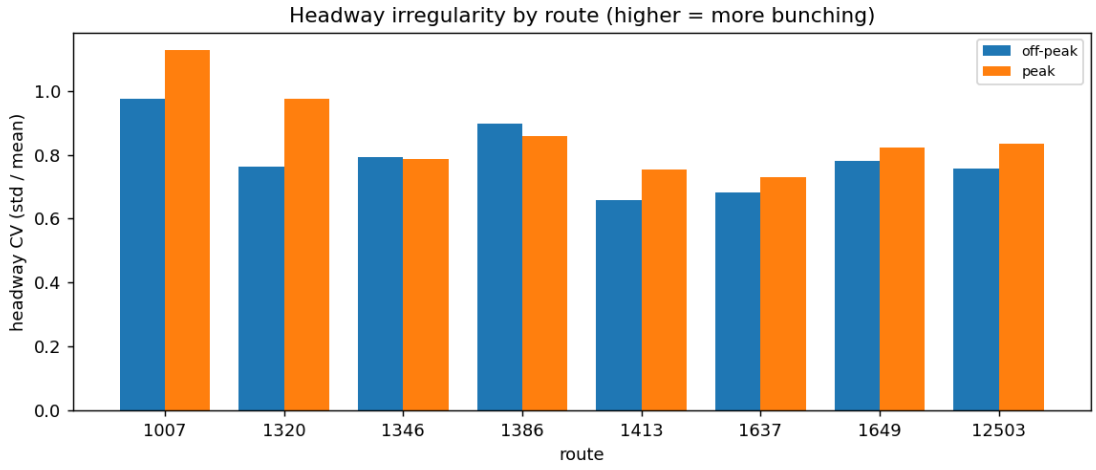

Figure 8: Within-window headway coefficient of variation per route, peak versus off-peak. Higher is more irregular; the peak is consistently worse.

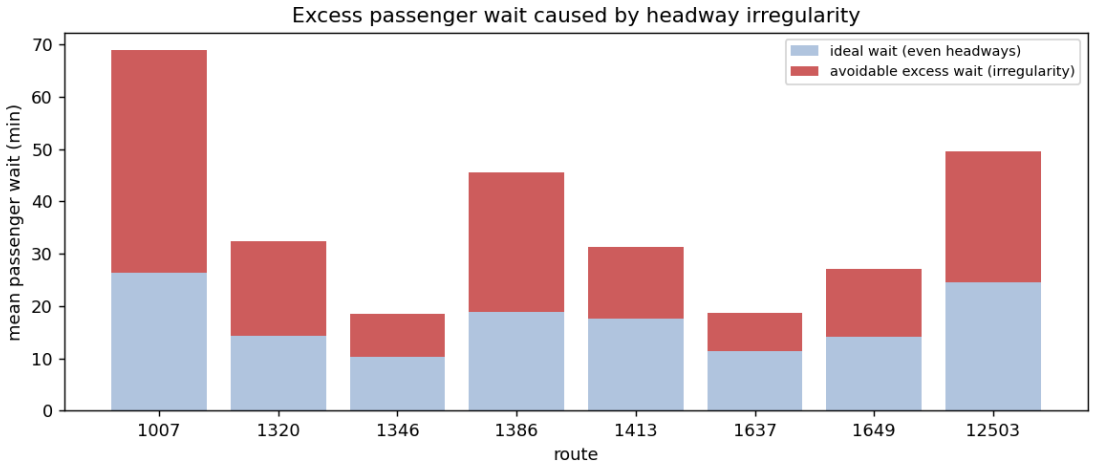

Figure 9: Mean passenger wait per route decomposed into the even-headway ideal ($\bar{h}/2$) and the *avoidable* excess caused by irregularity (Eq. 2).

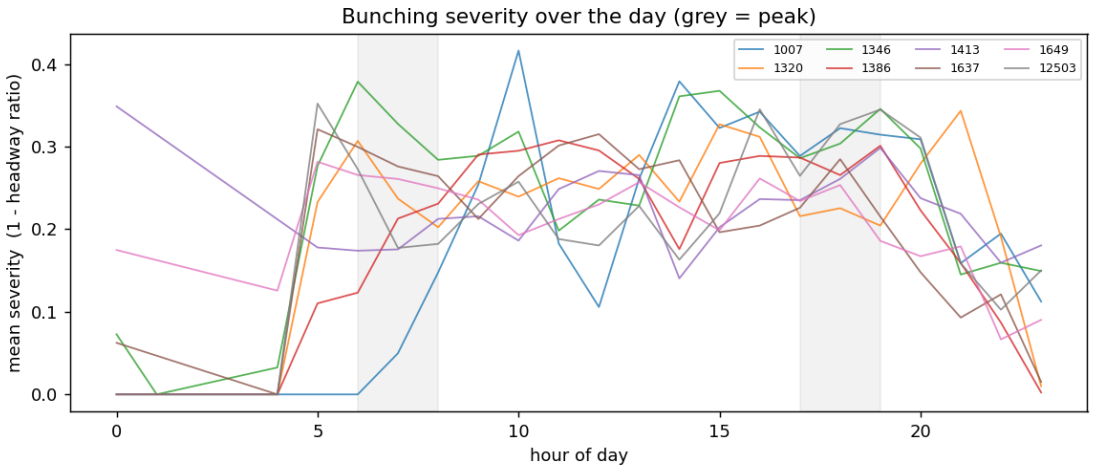

Figure 10: Mean bunching severity ($\max(0, 1 - \text{headway ratio})$) by hour of day, per route; shaded bands are the morning and evening peaks.

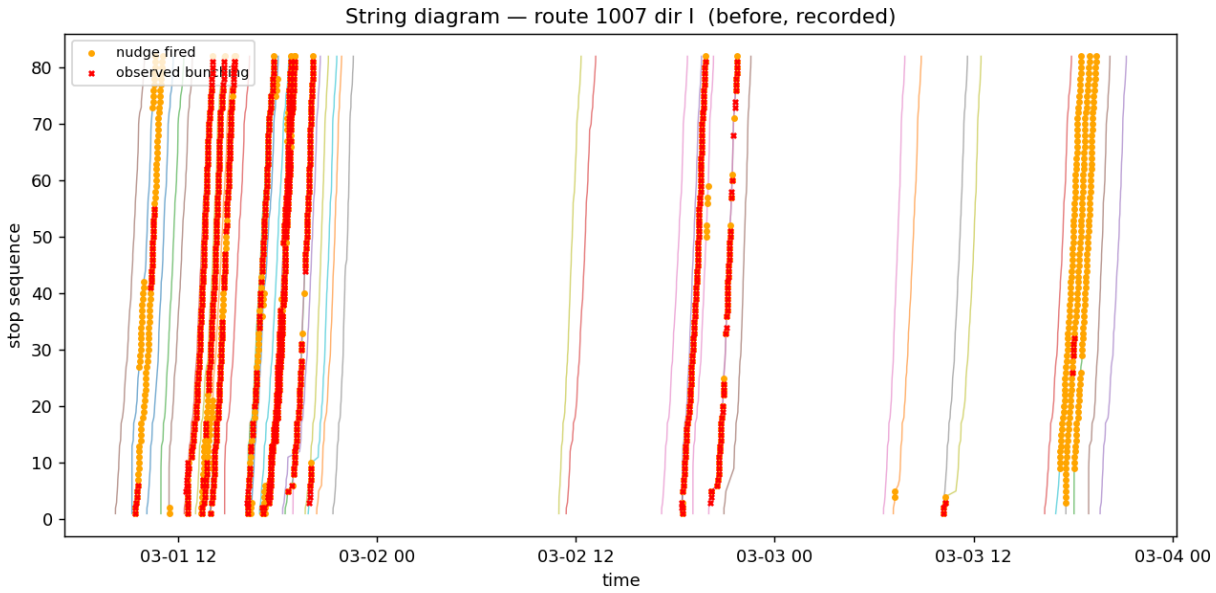

Figure 11: Per-bus trajectories ("string diagram") for one route reconstructed from raw OD. Each line is one bus over time; converging or crossing lines are bunching events. Orange/red markers show fired nudges and observed bunching. The signal exists because OD keeps individual vehicles and exact timestamps; aggregation into per-interval counts destroys it.

5.2 After: the same measurements, after the nudge

We now re-run the six "before" measurements of Section 5.1 on the simulated "after" arrival times (the counterfactual of Section 3.6). The subsections below mirror Section 5.1 one-to-one, each stating whether that measurement improved; Table 9 summarises the verdicts and Table 10 gives the full numeric panel. Under the default ease-off (8 s per nudged stop), the simulation rests on ≈11,000 small ease-offs, and two robustness checks support it: an *ablation* that gates the ease-off to only the model's *correct* (true-positive) nudges yields virtually the same reduction (−12.7% bunching / −9.8% wait), so the benefit is driven by the model's predictions rather than hindsight; and a *sensitivity sweep* of the ease-off magnitude from 4 to 16 s leaves the reduction in a narrow 12–13% band (Fig. 13).

Table 9: Each "before" measurement (Section 5.1) and its matching "after" subsection (Section 5.2), with the verdict.

Before	After	Verdict
Headway irregularity (CV)	§5.2.1	improves (−3.5%)
How close the buses run (ratio)	§5.2.2	improves (−11.2% / −12.6%)
Avoidable passenger waiting	§5.2.3	improves (−9.8%)
Severity over the day	§5.2.4	improves
Seeing bunching (string diagram)	§5.2.5	improves
Travel-time context	§5.2.6	not a target (unchanged)

5.2.1 Headway irregularity (CV) — improves

Mirrors Section 5.1.1. The mean headway CV falls 0.796 → 0.768 (−3.5%; middle panel of Fig. 12). The improvement is modest because the nudge removes the short-gap bunching it targets but leaves the long late-evening gaps, which dominate the raw spread, untouched.

Table 10: Before (actual March 2024) versus after (counterfactual nudge), from ≈11,000 ease-offs. "Improvement" is the reduction from before to after. The lower block lists indicators that the nudge is not designed to move (it evens spacing, it is not a speed-up).

Indicator	Before	After	Improvement
Improves with the nudge			
Bunching events (count)	21,123	18,463	−12.6%
Bunching rate (% of arrivals)	19.4%	17.2%	−11.2%
Avoidable wait (passenger-min)	201,978	182,196	−9.8%
Headway irregularity (mean CV)	0.796	0.768	−3.5%
Checked, but no improvement (the nudge is not a speed-up)			
Mean headway deviation (min)	25.4	25.1	−1.0%
Median wait (min)	10.7	10.7	≈ 0
Wait spread (std, min)	33.9	34.1	+0.4%

5.2.2 How close the buses run — improves

Mirrors Section 5.1.2. The share of arrivals that are bunched falls from 19.4% to 17.2% (−11.2%), and the raw count of bunching events from 21,123 to 18,463 (−12.6%; left panel of Fig. 12). This is the headline result: about one bunching event in eight does not form, because a closing bus eased off and let the gap reopen.

5.2.3 Avoidable passenger waiting — improves

Mirrors Section 5.1.3. Avoidable wait drops from 201,978 to 182,196 passenger-minutes (−9.8%; right panel of Fig. 12). The *median* wait itself barely moves (≈10.7 min before and after), because easing a closing bus off slightly delays it; the benefit is concentrated in the avoidable, variance-driven portion of the wait rather than the average (Section 6 develops this point).

5.2.4 Severity over the day — improves

Mirrors Section 5.1.4. The "after" severity curve sits at or below the "before" curve at every hour (Fig. 14), with the largest gap during the commute peaks. Severity falls automatically when there are fewer closed-up arrivals, so this is the same effect as the bunching reduction, now seen across the day.

5.2.5 Seeing bunching directly (string diagram) — improves

Mirrors Section 5.1.5. On the counterfactual arrival times fewer bus-lines converge into platoons than in the recorded diagram (Fig. 15 versus Fig. 11). The change is visible but subtle, as expected for a roughly one-in-eight reduction rather than a total fix.

5.2.6 Travel-time context — not a target

Mirrors Section 5.1.6. Unchanged by design. The nudge never speeds a bus up, so terminal-to-terminal ride time is not affected and the peak-versus-off-peak congestion of Table 8 is untouched. We report it only so the before/after pairing is complete, not as a claimed improvement.

6 Discussion

The results section established three things in turn: bunching is prevalent and expensive on these routes (Section 5.1), an Adaptive Random Forest predicts it with the best imbalance-aware skill and,

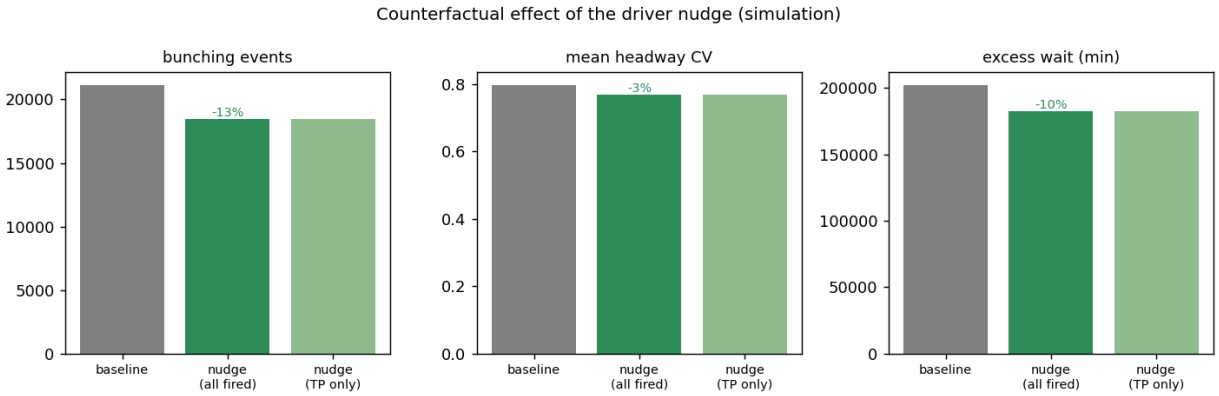

Figure 12: Counterfactual effect of the nudge (simulation): bunching events, headway CV, and avoidable wait, for the baseline versus the nudge applied on all fired warnings and on true-positive warnings only.

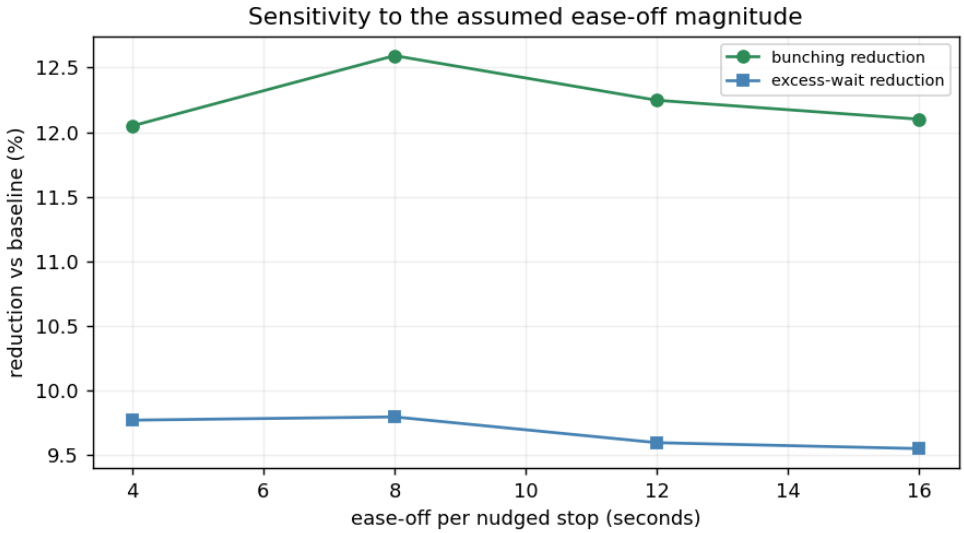

Figure 13: Sensitivity of the simulated reduction in bunching and in excess wait to the assumed per-stop ease-off magnitude. The effect is stable across 4–16 s.

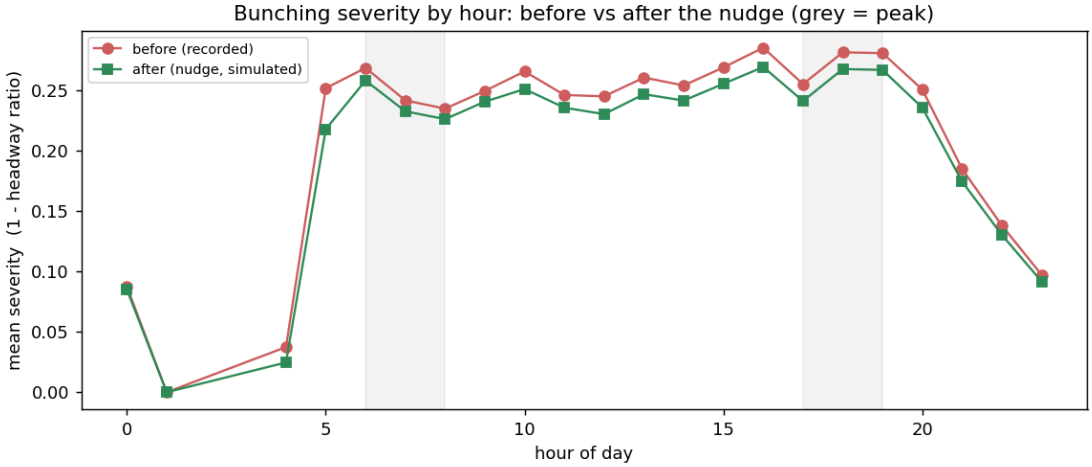

Figure 14: Bunching severity by hour, before (recorded) versus after (simulated nudge). The "after" curve sits at or below the "before" curve all day; shaded bands are the morning and evening commute peaks.

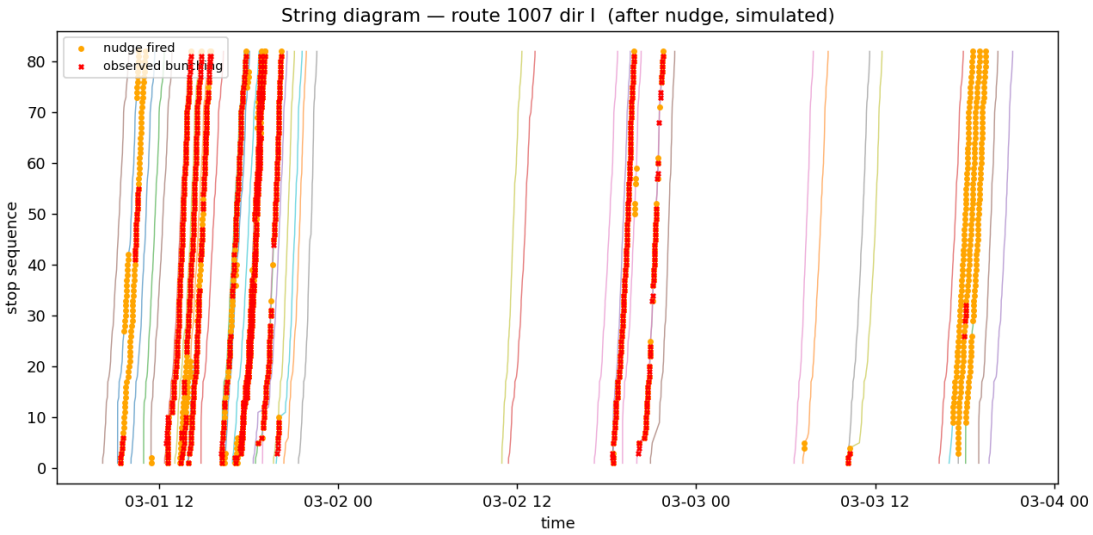

Figure 15: Per-bus trajectories on the simulated "after" arrival times (same route as Fig. 11). Fewer lines converge once closing buses ease off.

decisively, with actionable lead time (Section 4.1), and acting on those predictions in simulation reduces bunching by ≈13% and avoidable waiting by ≈10% (Section 5.2). This section interprets *why* the gains take the shape they do, how the approach relates to prior anti-bunching work, and what would have to hold for the estimated benefit to survive deployment.

Skill is not enough; lead time is the deciding factor. The adaptive ensemble gives the best skill on the rare bunching class while staying calibrated across the stream, but the more important observation is comparative. The static threshold is *competitive on F1*, so on a conventional accuracy metric the machine-learning model looks only marginally better. The advantage is concentrated in *balanced accuracy* and, decisively, in *lead time*: the threshold flags bunching only once the gap is already small, which is too late for a driver to act, whereas the streaming model warns minutes ahead. For an intervention that depends on a driver having time to ease off, a predictor that fires early is worth more than one that scores a fraction of an F1 point higher but fires late. This is why we treat lead time, not F1 alone, as the metric that decides whether a model can support a nudge.

Why the benefit is variance-driven, not a faster average. The before/after panel (Table 10) has a characteristic shape: the counts of bunching and the *avoidable* (variance-driven) wait fall, while the *median* wait, the mean headway deviation, and ride time barely move. This is the expected signature of the intervention rather than a disappointment. Easing a closing bus off slightly *delays* it, so a few passengers wait a little longer; what improves is the regularity of the spacing, not its speed. Equation (1) makes the mechanism explicit: mean wait is $\frac{\bar{h}}{2}(1 + \mathrm{CV}^2)$, so a reduction in CV removes the variance term $\frac{\bar{h}}{2}\mathrm{CV}^2$ while leaving the even-service floor $\frac{\bar{h}}{2}$ untouched. The nudge therefore trades a small, bounded mean delay for a larger reduction in the variance-driven excess. Read this way, a flat median wait alongside a falling avoidable wait is internal evidence that the simulation is behaving as the renewal model predicts, not against it.

Relation to prior anti-bunching approaches. The result sits within an established lineage but occupies a deliberately different point in it. A predictive line forecasts bunching without acting on it: Yu et al. (2016) predict stop-level headway and flag events several stops ahead, and our streaming classifier is a close relative that additionally closes the loop with an action. A control line acts but usually from the centre: Moreira-Matias et al. (2016) couple online prediction with automatically deployed holding, whereas our contribution keeps the prediction but replaces centralised control

with a decentralised, advisory ease-off. The ease-off itself is the advisory analogue of the classical "slow the leader, never the follower" control of Daganzo (2009) and Daganzo and Pilachowski (2011): we approximate their cooperative speed adjustment with a voluntary nudge to a single driver rather than an enforced hold. The modest size of the simulated wait saving is consistent with this literature rather than at odds with it: even sophisticated predictive control reports single-digit average wait reductions (Moreira-Matias et al., 2016), which places our ≈10% estimate in a credible range and suggests the gain is not inflated by the simulation.

What would have to hold in deployment. The estimated benefit assumes drivers act on the nudge, and that assumption is where a field deployment is most exposed. Phillips et al. (2015) find that the benefit of real-time control is robust to *random* partial non-compliance — much of the gain survives when only a fraction of advisories are followed — but degrades sharply when particular drivers *systematically* ignore the system. The practical implication is that a rollout should worry less about the occasional skipped nudge and more about persistent non-compliance by specific drivers, which points toward driver engagement and training rather than tighter enforcement. The advisory, zero-infrastructure design is well suited to that: because the only actuator is a voluntary change in one driver's pace, a single ignored nudge affects only that bus, and the system degrades gracefully rather than failing.

What this does and does not establish. The counterfactual is a simulation under stated assumptions — that nudged drivers comply, that a bounded ease-off is the right behavioural model, and that the local normal is fixed — so it is an *estimate* of plausible benefit, not a measured field outcome. What it does establish, more strongly than a static variance comparison would, is that a small, safe, decentralised action driven by the model's own predictions, and stable across the assumed ease-off magnitude, moves the headway geometry in the right direction.

What we deliberately do not claim. We do not claim a crowding benefit. In the SUNT data, bus occupancy barely tracks the headway ($r \approx 0.09$) and stop boardings do not rise with the gap ahead of a bus ($r \approx 0.00$), so there is no measurable spacing-driven crowding effect for the nudge to improve; we report crowding as unchanged rather than inventing one. Nor do we claim to address congestion: the longer peak ride times are ordinary road congestion, and the nudge evens *spacing* rather than adding speed, so it neither should nor does change ride time.

Local feasibility (Salvador) SUNT OD is sufficient for a research prototype and demo. A real in-cab deployment would require live positioning feeds and an in-cab device across multiple public and private operators; the copilot is a low-cost behavioural aid, not a control system or infrastructure.

Limitations Several limitations bound these results. Bunching is threshold-defined and sensitive to that definition (which is why the evidence of Section 5.1 is windowed and the counterfactual is swept over the ease-off magnitude); the class is imbalanced; about 23% of trips are dropped for non-monotonic timestamps at this scope; onboard loading is partly inferred upstream in SUNT; we scope to the eight busiest routes; and the real effect of drivers easing off cannot be observed from historical data — the "would bunching fall?" estimate of Section 5.2 is a simulation under stated assumptions (most consequentially driver compliance, discussed above), not a measured outcome. A further modelling limitation is the labelling reference: our fixed local-normal headway implicitly treats the recent fleet as behaviourally homogeneous, whereas Bartholdi III and Eisenstein (2012) show a fair headway can emerge endogenously rather than being fixed in advance, and Martínez-Estupiñan et al. (2023) show that drivers differ significantly in speed and that ignoring this misstates a control tool's benefits. A per-driver or self-equalising reference headway is therefore a natural refinement.

Safety, equity, privacy The nudge must be glanceable or audio-only, advisory, and never instruct the driver to speed up; the safe action space is ease-off or a brief hold. We include high-frequency feeder routes that serve peripheral, lower-income areas, where bunching hurts most. The pipeline uses vehicle- and stop-level aggregates only, with no passenger identification or tracking. We do not claim to "solve Salvador traffic": this is a reproducible demonstration of streaming anti-bunching nudges on selected routes.

7 Conclusion

We presented a driver-facing anti-bunching copilot that predicts next-window bunching risk from SUNT OD-derived forward-headway signals using online CapyMOA learners under prequential evaluation. An Adaptive Random Forest gives the best imbalance-aware skill and, unlike non-learning baselines, warns the driver with actionable lead time. The approach eases bunching behaviourally with no infrastructure, is reproducible from the submitted code, and is framed honestly as an advisory aid rather than a traffic-control system.

Data and code availability

The SUNT dataset is public: code and documentation at `https://github.com/LabIA-UFBA/SUNT`; processed data (incl. OD) at `https://huggingface.co/datasets/labiaufba/PublicTransportationSunt`; raw sources at `https://data.mendeley.com/datasets/85fdtx3kr5/1`; dataset paper at `https://doi.org/10.1038/s41597-025-05674-6`. The experiment code (data preparation, prequential CapyMOA pipeline, and figure generation) is included with this submission and is fully reproducible.

References

Gabriel Aguiar, Bartosz Krawczyk, and Alberto Cano. A survey on learning from imbalanced data streams: taxonomy, challenges, empirical study, and reproducible experimental framework. *Machine Learning*, 113(7):4165–4243, 2024. doi: 10.1007/s10994-023-06353-6.

Banco Nacional de Desenvolvimento Econômico e Social (BNDES) and Ministério das Cidades. Relatório de diagnóstico: Região metropolitana de salvador. Relatório de Diagnóstico Volume 4, Estudo Nacional de Mobilidade Urbana, February 2025. URL `https://www.bndes.gov.br/arquivos/fep-mobilidade-urbana/relatorios-diagnostico/relatorio-Salvador-vol-4.pdf`. Accessed 18 June 2026.

John J. Bartholdi III and Donald D. Eisenstein. A self-coördinating bus route to resist bus bunching. *Transportation Research Part B: Methodological*, 46(4):481–491, 2012. doi: 10.1016/j.trb.2011.11.001.

Albert Bifet and Ricard Gavaldà. Learning from time-changing data with adaptive windowing. In *Proceedings of the 2007 SIAM International Conference on Data Mining (SDM)*, pages 443–448, 2007. doi: 10.1137/1.9781611972771.42.

Albert Bifet and Ricard Gavaldà. Adaptive learning from evolving data streams. In *Proceedings of the 8th International Symposium on Intelligent Data Analysis (IDA)*, pages 249–260, 2009. doi: 10.1007/978-3-642-03915-7_22.

Veruska Borges Santos, Carlos Eduardo S. Pires, Dimas Cassimiro Nascimento, and Andreza Raquel M. de Queiroz. A decision tree ensemble model for predicting bus bunching. *The Computer Journal*, 65(8):2044–2062, 2022. doi: 10.1093/comjnl/bxab045.

CapyMOA Developers. CapyMOA: Machine learning for data streams in Python. `https://capymoa.org/`, 2024.

Carlos F. Daganzo. A headway-based approach to eliminate bus bunching: Systematic analysis and comparisons. *Transportation Research Part B: Methodological*, 43(10):913–921, 2009. doi: 10.1016/j.trb.2009.04.002.

Carlos F. Daganzo and Josh Pilachowski. Reducing bunching with bus-to-bus cooperation. *Transportation Research Part B: Methodological*, 45(1):267–277, 2011. doi: 10.1016/j.trb.2010.06.005.

Pedro Domingos and Geoff Hulten. Mining high-speed data streams. In *Proceedings of the 6th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 71–80, 2000. doi: 10.1145/347090.347107.

Fatemeh Enayatollahi, Ahmed Osman Idris, and M. A. Amiri Atashgah. Modelling bus bunching under variable transit demand using cellular automata. *Public Transport*, 11(2):269–298, 2019. doi: 10.1007/s12469-019-00203-2.

Marcos V. Ferreira, Matheus Souza, Tatiane N. Rios, Islame F. C. Fernandes, Jorge Nery, João Gama, Albert Bifet, and Ricardo A. Rios. Salvador urban network transportation (SUNT): A landmark spatiotemporal dataset for public transportation. *Scientific Data*, 12(1):1320, 2025. doi: 10.1038/s41597-025-05674-6.

Heitor M. Gomes, Albert Bifet, Jesse Read, Jean Paul Barddal, Fabrício Enembreck, Bernhard Pfharinger, Geoff Holmes, and Talel Abdessalem. Adaptive random forests for evolving data stream classification. *Machine Learning*, 106(9):1469–1495, 2017. doi: 10.1007/s10994-017-5642-8.

Yerly Martínez-Estupiñan, Felipe Delgado, Juan Carlos Muñoz, and Kari E. Watkins. Improving the performance of headway control tools by using individual driving speed data. *Transportation Research Part A: Policy and Practice*, 174:103761, 2023. doi: 10.1016/j.tra.2023.103761.

Luís Moreira-Matias, Oded Cats, João Gama, João Mendes-Moreira, and Jorge Freire de Sousa. An online learning approach to eliminate bus bunching in real time. *Applied Soft Computing*, 47: 460–482, 2016. doi: 10.1016/j.asoc.2016.06.031.

Duy Q. Nguyen-Phuoc, Graham Currie, Chris De Gruyter, Inhi Kim, and William Young. Modelling the net traffic congestion impact of bus operations in Melbourne. *Transportation Research Part A: Policy and Practice*, 117:1–12, 2018. doi: 10.1016/j.tra.2018.08.005.

William Phillips, Andrés del Rio, Juan Carlos Muñoz, Felipe Delgado, and Ricardo Giesen. Quantifying the effects of driver non-compliance and communication system failure in the performance of real-time bus control strategies. *Transportation Research Part A: Policy and Practice*, 78:463–472, 2015. doi: 10.1016/j.tra.2015.06.005.

Shengjie Qiang and Qingxia Huang. Impacts of bus holding strategies on the performance of mixed traffic system. *Physica A: Statistical Mechanics and its Applications*, 611:128455, 2023. doi: 10.1016/j.physa.2023.128455.

TomTom International B.V. Traffic index ranking 2025. TomTom Traffic Index, 2025a. URL `https://www.tomtom.com/traffic-index/ranking`. Accessed 18 June 2026.

TomTom International B.V. Salvador traffic report. TomTom Traffic Index, 2025b. URL `https://www.tomtom.com/traffic-index/city/salvador`. 2025 traffic data; accessed 18 June 2026.

Tribunal de Contas do Estado da Bahia. Auditoria: Compensação tarifária do sistema metroviário de salvador e lauro de freitas (SMSL). Technical Report 25, Tribunal de Contas do Estado da Bahia, Salvador, Brazil, 2022. URL `https://www.tce.ba.gov.br/files/com_cdspublicacaoinstitucional/publicacoes/arquivo/Sumario-2022CompTarifSistMetroviario-REVISAO-10nov2216h412022102420231045.pdf`. Accessed 18 June 2026.

Shuo Wang, Leandro L. Minku, and Xin Yao. Resampling-based ensemble methods for online class imbalance learning. *IEEE Transactions on Knowledge and Data Engineering*, 27(5):1356–1368, 2015. doi: 10.1109/TKDE.2014.2345380.

World Bank. Brazil electromobility multiphase programmatic approach—MPA phase 2 salvador (P507029). Project Information Document, Appraisal Stage PIDIA01274, World Bank, Washington, DC, February 2026. URL `https://documents1.worldbank.org/curated/en/099021026100037052/pdf/P507029-8e092aac-8da9-4b2e-8d67-bc039993244e.pdf`. Prepared or updated 10 February 2026; accessed 18 June 2026.

Haiyang Yu, Dongwei Chen, Zhihai Wu, Xiaolei Ma, and Yunpeng Wang. Headway-based bus bunching prediction using transit smart card data. *Transportation Research Part C: Emerging Technologies*, 72:45–59, 2016. doi: 10.1016/j.trc.2016.09.007.