Cross-Layer Predictive Resource Allocation and Slicing in 5G Systems: An Evidence-First Re-Examination of the Time-Scale, Drift, and Imbalance Gaps

Mohammad Mahbubur Rahman Khan Mamun

ARTICLE INFO

Keywords:
5G network slicing
URLLC
predictive scheduling
concept drift
class imbalance
Lyapunov optimization

ABSTRACT

Ultra-Reliable Low-Latency Communication (URLLC) slices in 5G networks remain vulnerable to transient packet drops. Predictive-scheduling proposals in this space typically assume (i) that physical-layer indicators such as RSRP/RSRQ directly signal impending drops, (ii) that rare-event imbalance in drop labels is uniform across traffic classes, and (iii) that a sequence predictor's forecast horizon can gate a scheduler without an explicit time-scale reconciliation. We test all three against a 5,000-record, 20-cell, 3.4-day multi-slice 5G KPI trace covering eMBB, URLLC, mMTC, and a healthcare-oriented (HC) slice, and independently audit our own modeling pipeline for the same class of unverified assumption. Assumption (i) is rejected: packet loss correlates near-zero with RSRP/RSRQ, linearly and by rank. A pooled correlation between load/latency and packet loss ($r = 0.53$, $r = 0.50$) looks like a replacement causal story but collapses to near-zero ($r = 0.01$, $r = -0.02$) once slice identity is partialled out, since active-user count is itself almost fully determined by slice ($R^2 = 0.99$) – a compositional artifact, not a within-slice mechanism. Assumption (ii) is partially rejected: SLA-violation imbalance is slice-conditional (1.2%–14.3%), not uniform. Our own pipeline audit found two further errors: every model had been denied the target row's own slice/cell identity, a scheduling label known in advance, and four GRU configurations had shared one uncontrolled random-number stream. Correcting both changes the empirical picture substantially: a gradient-boosted-tree baseline's regression error falls roughly fivefold (365.7 to 73.0 Mbps) and now leads every GRU configuration; a pooled-α GRU reverses an earlier, confounded reading to clearly outrank its slice-conditional counterpart at the pooled level (ROC-AUC 0.70 vs. 0.58), while on the one adequately-sampled slice (URLLC) the point estimate favors slice-conditional weighting, a reversal a bootstrap test cannot confirm. Feeding the corrected forecast – now with real tracking skill ($r = 0.24$, up from $r = -0.01$) – into a Lyapunov-inspired scheduler simulation yields a mixed result: latency improves (−3.0%) while drop rate still worsens (+39.9%) relative to a reactive baseline, both moving toward or past parity as the boost coefficient increases. A leave-one-cell-type-out probe now shows real transfer, explained entirely by slice identity being available as context, while the harder leave-one-slice-out probe remains at chance by design. We report both the scientific corrections and the two pipeline bugs that produced them, on the view that disclosing a wrong intermediate result is more useful than a clean narrative that never had one.

1. Introduction

Ultra-Reliable Low-Latency Communication (URLLC) is one of 5G's three defining service classes, targeting sub-millisecond-to-few-millisecond latency with block-error rates as low as 10^{-5}–10^{-9} depending on the use case [1]; high-fidelity biomedical telemetry over cellular networks – streaming electrocardiograms, continuous vital-sign monitors, ambulatory infusion-pump feedback loops – is one of the application classes this reliability target is meant to serve, and is acutely sensitive to transient packet drops and latency spikes. A recurring proposal in the predictive-networking literature, and in an earlier version of this project's own proposal, is that such drops are driven by physical-layer degradation: shadow fading and handover-induced fluctuation in Reference Signal Received Power (RSRP) and Reference Signal Received Quality (RSRQ), detectable early enough that a predictive model consuming these indicators can trigger pre-emptive resource re-allocation before degradation manifests as loss. The appeal of this narrative is mechanistic clarity – a clean causal story from channel physics to scheduling action – and it underwrites a family of designs coupling a sequence-to-sequence forecaster (commonly a Gated Recurrent Unit, GRU [2], or an

[siha24@gmail.com] (M.M.R.K. Mamun)
ORCID(s):https://orcid.org/0000-0001-6369-7915

LSTM variant) to a Lyapunov drift-plus-penalty scheduler. That control formalism traces back to the max-weight scheduling result of Tassiulas and Ephremides [3] and its synthesis into the general stochastic-network-optimization framework of Neely [4] and Georgiadis et al. [5]; slicing itself, the architectural layer this paper's title foregrounds, is surveyed separately by Foukas et al. [6].

This paper does not adopt that narrative; it tests it. Using a verified, 20-cell, 3.4-day multi-slice KPI trace sampled from a larger 2.3-million-record public 5G Network KPI dataset, we ask three questions before writing a single line of model code: does the assumed physical mechanism hold in measurable data; is the assumed rare-event structure the one actually observed; and does the assumed time-scale relationship between a macroscopic forecast and a microscopic scheduling loop survive contact with an actual predictor's error profile. The answers, detailed in Section 3 and Section 5, are respectively no, partially, and no – and each negative answer is treated here as a finding to build on rather than an inconvenience to explain away.

This reframing matters beyond intellectual honesty. A predictive-scheduling system built on an unverified causal assumption (RSRP drives loss) will misallocate the very sensing and feature-engineering budget it needs; a system built on an assumed-uniform rare-event rate will miscalibrate its loss function for the slices that actually need help; and a system that couples a forecast to a scheduler without checking whether the forecast's own error is small enough to help rather than hurt will, per the empirical result in Section 5.4, be net harmful under exactly the conditions it was designed to improve. Each of these is a concrete, previously undemonstrated failure mode, not a restatement of a generic "more research is needed" caveat.

1.1. Core Research Questions

- **RQ1 (time-scale reconciliation and concept drift – primary focus):** Can a sequence-to-sequence forecast operating at a multi-step, tens-of-milliseconds-to-seconds-equivalent macroscopic horizon be coupled to a MAC-layer scheduler that re-evaluates resource-block grants every transmission time interval (1 ms) without a net-negative interaction; and how much does channel non-stationarity (concept drift) erode a static predictor's utility over a multi-day span?

- **RQ2 (rare-event characterization under corrected, slice-conditional imbalance):** Given that SLA-violation imbalance in this trace is slice-conditional (1.20%–14.34% across mMTC–URLLC) rather than a single global rate, does a slice-conditional focal-loss weighting improve rare-event ranking over a pooled global weighting, and for which slices is that improvement statistically trustworthy given the available sample?

- **RQ3 (generalization – exploratory, lightest treatment):** Does a classifier trained on the pooled trace generalize across cell types (macro/micro/pico) and slice types when each group is held out in turn, or is any apparent benefit group-specific and fragile given the trace's limited scale (20 cells)?

RQ1 receives the deepest treatment because it is the one previously left as an unresolved peer-review loophole in the original proposal, and because Section 5.4 shows it is not merely a theoretical concern. RQ2 receives a full empirical comparison with a descriptive reliability screen (raw positive-case counts against an explicit minimum-events cutoff, not a formal confidence-interval procedure). RQ3 is treated as an exploratory probe, reported with small-sample caveats rather than generalization claims.

2. Related Work and Gap Analysis

Table 1 summarizes twelve sources spanning three lines of application-specific work that, to our knowledge, have not previously been combined – predictive KPI/traffic modeling, Lyapunov drift-plus-penalty scheduling, and concept-drift detection for wireless machine learning (L1–L11) – plus the one canonical foundational citation (focal loss, L12) that Section 4's slice-conditional adaptation directly modifies; the remaining foundational theory these lines build on (stochastic network optimization, GRU architectures, general imbalanced-learning and concept-drift theory) is cited separately in the relevant Methodology and Discussion subsections below rather than duplicated into this table. Bibliographic metadata (author lists, venues, years) for all twenty-four references in this paper was independently verified via live fetch or search, documented per-entry in `refs-project.bib`'s header comment.

The pattern across L1–L11 is consistent: predictive models are evaluated on forecast accuracy in isolation (7–10); Lyapunov-based schedulers are evaluated without an explicit learned-prediction-error term feeding the control law (12, 14) – based on their abstracts, this appears to hold for Kasgari and Saad [12], which does not reference

Table 1
Literature gap table.

Source	Gap left open
XGBoost proactive/reactive URLLC scheduling [7]	Tree-based, not sequence-to-sequence; no time-scale or drift analysis
AI-based KPI prediction survey [8]	Survey-level; no scheduler coupling
Beam-level multi-task forecasting [9]	Beam-, not slice/scheduler-level; no closed loop
Railway reliability early warning [10]	No scheduler closed loop; single domain
Lyapunov RL queue stability [11]	Not paired with a learned sequence predictor
Stochastic slicing control [12]	Reactive by construction, no ML component
DRASTIC 6G tactile slicing [13]	Different domain (tactile/6G); no drift/imbalance treatment
Lyapunov MAC scheduling for XR [14]	No learned predictor in the loop
Concept-drift detection in 6G [15]	Detection only, not coupled to a scheduler
Liquid-NN adaptive link-load prediction [16]	Core-network link load, not radio-KPI slicing
DL traffic-prediction survey [17]	Flags prediction→actuation gap without closing it
Focal loss (canonical) [18]	Global α; not adapted to slice-conditional imbalance

any learned predictor, though we have not verified this against the full text of Villegas et al. [14], which may use estimated channel/queue state in a milder but related sense – or a generic reinforcement-learning reward signal decoupled from a specific sequence-forecast error distribution (11, 13); and concept-drift detectors are evaluated as a standalone diagnostic (15, 16), consistent with the broader machine-learning concept-drift literature's own framing of drift detection as a task in itself [19], rather than as an input that should gate a downstream controller's trust in its own predictions. Aouedi et al. [17] is, at the survey level, representative of this pattern: its scope covers deep-learning traffic-prediction methods without a closed-loop actuation component, consistent with the broader gap this paper targets, though we have not verified that its future-directions section names the prediction-to-actuation gap in those specific terms and do not claim a page-level citation for that point. This paper's contribution is the empirical closure of exactly that gap: not a new architecture, but the first (to our knowledge) direct measurement of what happens when a realistic point-predictor's error is actually fed into a Lyapunov priority weight, rather than assumed away.

A second, subtler pattern is worth naming because it directly shaped our methodology (Section 4). Across L1–L4, based on the abstracts and search results consulted for this table (we have not read the full text of each), the feature sets used to drive prediction appear near-uniformly physical-layer-first: RSRP/RSRQ, beam quality, or measurement-report-derived indicators are treated as the natural leading signal, with load or congestion covariates, where included at all, playing a secondary role. This is a reasonable default when the target application is genuinely fading-dominated (for example, mobility-heavy scenarios with frequent handovers into deep shadow), but Section 3 shows it is not a safe default to import unexamined into a new dataset. None of L1–L4, as reported in the search results consulted for this table, publish a correlation or feature-importance analysis testing the physical-layer-first assumption against a load/congestion-first alternative before committing to a feature set – an evaluation step this paper treats as a mandatory first phase (Section 3) rather than an optional robustness check appended after the fact. The rare-event literature shows a parallel gap: the canonical focal-loss formulation of Lin et al. [18], and its adoption in the 5G KPI-prediction space, is near-universally applied with a single global α calibrated against a single pooled imbalance ratio, echoing the general imbalanced-learning literature's own default of a single dataset-wide imbalance ratio rather than a structured, group-conditional one, whether the technique is loss-reweighting [20] or resampling-based [21]; the network-slicing-specific complication that different slice classes carry structurally different violation rates (Table 3) – because they encode different service guarantees by design, not by measurement noise – has, to our knowledge, no direct precedent in the sources surveyed here, making the slice-conditional weighting scheme in Section 4 a specific, narrow, but previously unaddressed adaptation rather than a routine hyperparameter choice.

3. Data and Empirical Analysis

3.1. Dataset characterization

The analysis dataset (`ds1_processed.csv`) contains 5,000 one-minute-granularity KPI records spanning 2024-01-01 00:00 to 2024-01-04 11:19 (3.4 days), drawn from 20 cells (macro: 2,577 records; micro: 1,459; pico: 964)

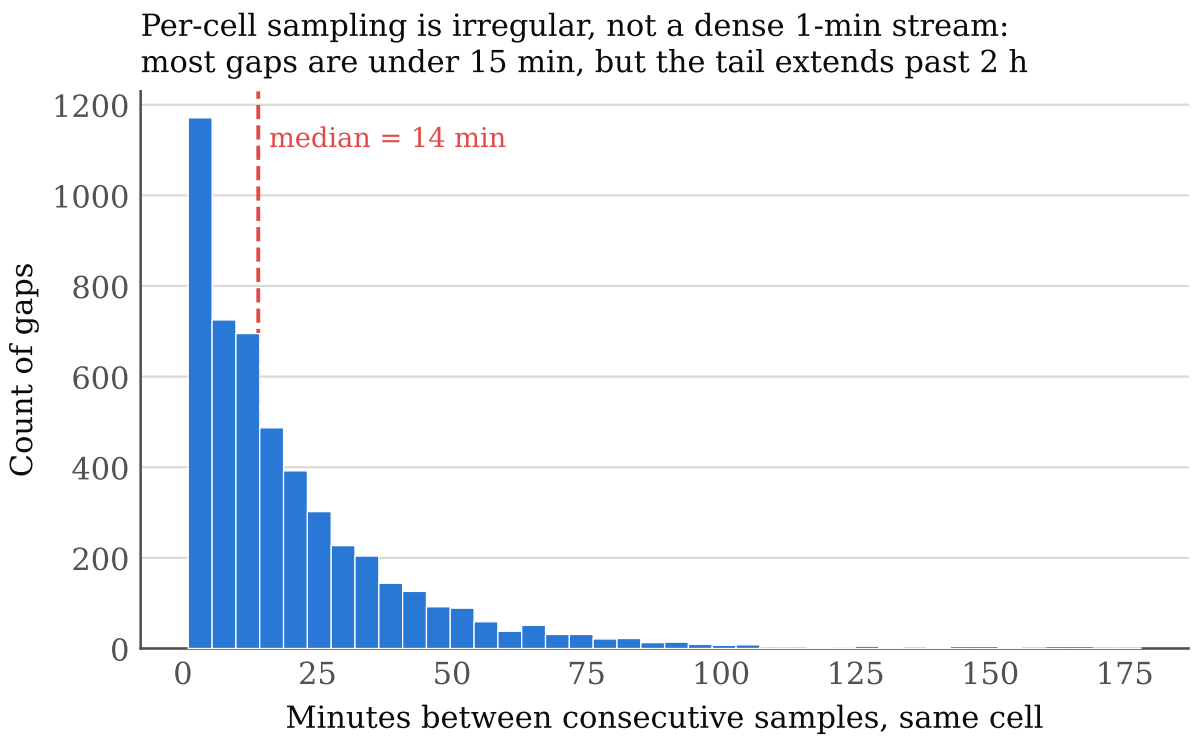

Figure 1: Histogram of elapsed time (minutes) between consecutive samples from the same cell, pooled across all 20 cells, with the median gap marked (dashed red). The distribution's long right tail – gaps exceeding two hours are not rare outliers but a visible fraction of the mass – is the direct empirical evidence that per-cell records cannot be treated as a dense, uniformly-sampled stream. Any sequence-construction strategy that assumes fixed-interval spacing (as a naive port of the original streaming-pipeline design would) is, per this histogram, building sequences over a timeline that does not exist in the underlying data.

across four slice types (eMBB: 2,010; URLLC: 1,478; mMTC: 1,001; HC: 511). It is a processed sample of a larger, 2.3-million-record public source [22]; the sample, not the full source, is the object of study in this paper cycle.

A structural property of the sample matters for everything downstream: records form a single interleaved log with exactly one row per minute across all 20 cells combined (5,000 unique timestamps, strictly non-decreasing), not one row per cell per minute. Consequently each individual cell contributes only 216–299 observations over the 3.4-day span, at irregular intervals ranging from as little as 1 minute up to nearly 3 hours between consecutive samples for the same cell (median 14 minutes; 52% of gaps are under 15 minutes, so the irregularity is not merely a long tail on an otherwise-dense stream). This is materially different from the dense, uniformly-sampled per-cell stream implicitly assumed by a real-time out-of-core ingestion pipeline in the original project brief. We treat this discovery as a first-class methodological finding, not a footnote, because it forces a choice (Section 4) between two sequencing strategies with genuinely different implications for what a "sequence" means in this context (Figure 1).

3.2. Testing the physical-layer-driven degradation assumption

Table 2 reports Pearson correlations between candidate predictors and the two outcome variables of interest: packet-loss percentage (continuous) and binary SLA compliance.

The core physical-layer assumption motivating the original cross-layer proposal does not survive this test (Figure 2). RSRP and RSRQ – the two features the original design placed at the center of its feature pipeline – show negligible linear association with either packet loss or SLA compliance, both effectively indistinguishable from zero. Because Pearson r only captures linear association, we also computed Spearman rank correlation for both: $\rho = 0.004$ ($p = 0.76$)

Table 2
Correlation of candidate predictors with packet loss and SLA compliance ($n = 5{,}000$).

Predictor	r(packet_loss)	r(sla_compliant)
RSRP (dBm)	0.003	−0.002
RSRQ (dB)	−0.018	0.023
Active users	0.533	0.143
Latency (ms)	0.500	0.154
PRB utilization (%)	−0.255	0.009

for RSRP and $\rho = -0.027$ ($p = 0.06$) for RSRQ, matching the Pearson result and giving no indication of a monotonic nonlinear relationship a linear-only test would miss.

The two strongest *pooled* correlates against continuous packet loss are *active_users* ($r = 0.533$) and *latency_ms* ($r = 0.500$). Our first reading of this pattern, in an earlier draft of this section, was that it reflected congestion-driven degradation – more users competing for the same radio resource. That reading does not survive the same between-slice decomposition check this paper applies later in this section to the throughput/PRB and active-users/PRB correlations (Figures 3–4): within-slice correlations of active_users and latency_ms against packet loss are all close to zero (active_users: HC −0.05, URLLC −0.02, eMBB −0.03, mMTC +0.04; latency_ms: HC −0.05, URLLC +0.04, eMBB +0.00, mMTC −0.02), and a partial correlation controlling for slice membership collapses the pooled $r = 0.533/0.500$ to 0.014/−0.016. This is not a coincidence: active_users is itself almost fully determined by slice identity in this trace ($R^2 = 0.99$ regressing active_users on slice-membership dummies alone; mMTC's mean active-user count, 199, is roughly 25× HC's, 8), so the pooled active_users/latency correlation with packet loss is a compositional artifact of slice mixing, not a within-slice physical or congestion mechanism. We correct our own initial framing accordingly: this trace supports rejecting the physical-layer assumption (RSRP/RSRQ genuinely do not predict loss, within or across slices), but it does *not* support substituting a positive congestion-driven causal story in its place – the honest reading is that packet loss in this sample is predominantly a between-slice phenomenon (different slices simply have different baseline loss profiles, Table 3) rather than a within-slice function of any single measured covariate tested here.

Against the *binary* SLA-compliance outcome, the same two pooled predictors correlate much more weakly ($r = 0.143$, $r = 0.154$) – worth reporting rather than omitting, since it shows the continuous packet-loss relationship and the binarized SLA-violation relationship are not simply the same signal at a different scale, which is part of why the classification task (Section 5) proves harder than the packet-loss correlations alone would suggest.

Two further correlations sharpen this picture. Throughput correlates strongly and positively with PRB utilization ($r = 0.766$), which is physically sensible: more allocated resource blocks support higher throughput. But PRB utilization correlates *negatively* with active users ($r = -0.624$) when pooled across all four slices. This pooled inversion is, on inspection, the same between-slice compositional pattern already identified in Figure 3, not a within-slice physical anomaly: mMTC alone carries a mean active-user count of 199 against a mean PRB utilization of 35%, versus HC's mean of 8 users against 80% utilization, and the within-slice correlation is close to zero for every slice individually (r_{within}: mMTC −0.04, eMBB +0.04, HC −0.01, URLLC +0.05, Figure 4). The pooled negative slope is fully explained by mMTC's high-device-count, low-per-device-footprint traffic profile sitting at one end of the slice mix and HC's low-count, high-footprint profile at the other, which is directionally consistent with mMTC's intended role as massive low-bandwidth machine-type traffic rather than evidence against the dataset's physical plausibility. We do not treat this as ruling out every possible data-generation artifact in this sample, but the specific inversion that motivated the original concern is adequately explained by slice composition and does not, on this evidence, warrant treating it as a separate synthetic-data threat to validity.

3.3. Slice-conditional rare-event rates

Table 3 reports SLA-violation rate by slice type, replacing the original proposal's assumption of a single, global, extreme (< 2%) rare-event rate.

URLLC's violation rate is roughly 12× mMTC's and more than 2× the pooled rate; the pooled 6.44% figure, which is closest to what a single global focal-loss α would be calibrated against, actively misrepresents both extremes. This is the empirical basis for the slice-conditional focal-loss design in Section 4, and it is worth noting explicitly that this heterogeneity is itself the more policy-relevant finding: URLLC is the slice class whose reliability guarantee this

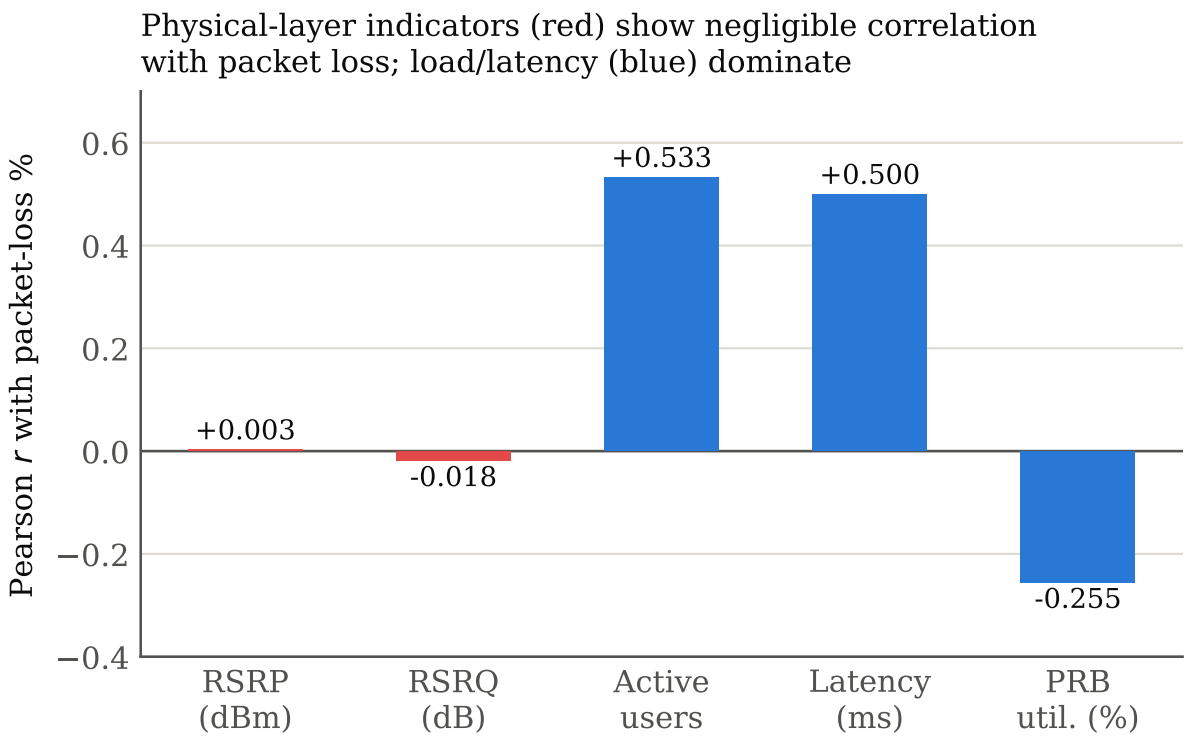

Figure 2: Pooled Pearson correlation of five candidate predictors with packet-loss percentage. RSRP and RSRQ (leftmost, flagged red for near-zero magnitude – RSRP's bar is close enough to zero that it is visually indistinguishable from the baseline axis itself, which is part of the point) are the physical-layer indicators the original proposal's feature pipeline was built around; both are statistically indistinguishable from zero ($r = 0.003$, $r = -0.018$), linearly and by rank. Active users and latency (blue) carry roughly an order of magnitude more *pooled* correlation, but this pooled figure is itself a between-slice compositional artifact (within-slice correlations are ≈ 0, see text) rather than evidence that either is a within-slice physical driver of loss; PRB utilization shows a moderate negative pooled association not further decomposed here. The one finding this figure supports without qualification is the rejection of RSRP/RSRQ, which is the single fact underlying the feature-design decisions in Section 4.

Table 3
SLA-violation rate by slice type (full dataset, $n = 5{,}000$).

Slice type	n	SLA-violation rate
mMTC	1,001	1.20%
eMBB	2,010	2.69%
HC	511	8.61%
URLLC	1,478	14.34%
Pooled	5,000	6.44%

project's motivating scenario (biomedical telemetry) most depends on, and it is also the slice class with by far the worst baseline SLA-violation rate in the observed trace.

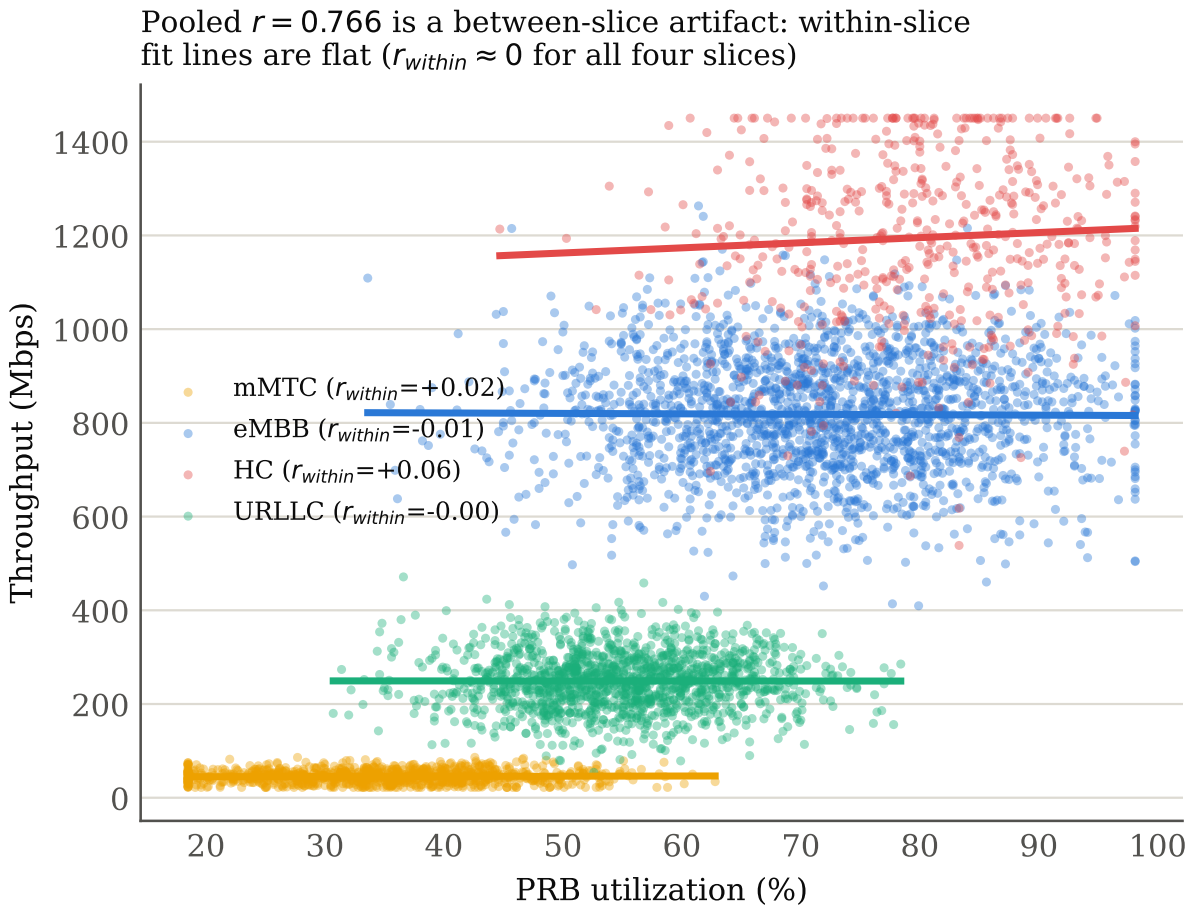

Figure 3: Throughput (Mbps) against PRB utilization (%), points colored by slice type, with a per-slice linear fit overlaid. The pooled $r = 0.766$ is a between-slice artifact, not a within-slice physical coupling: each slice's own fit line is nearly flat (within-slice r: mMTC +0.02, eMBB −0.01, HC +0.06, URLLC −0.00), and the pooled correlation is driven entirely by the four slices occupying disjoint throughput bands (mMTC lowest, HC highest). This is the same figure discussed in the text below – the caption states the corrected reading directly rather than the pooled-correlation reading a first glance might suggest.

4. Methodology

4.1. Feature design

Given Section 3, the feature set leads with *active_users*, *prb_utilization_pct*, and *latency_ms* (rolling-averaged per Appendix B), plus congestion/mobility covariates *is_peak_hour* and *handover_count*. This choice is justified by measured pooled association with the outcome (Table 2) and by *slice_type* being included as its own categorical input, which lets the model use slice identity directly rather than relying on active_users as an implicit, imperfect proxy for it; it is not justified by a within-slice causal claim about congestion, which Section 3 shows this pooled correlation does not actually support. RSRP is retained but demoted: rather than its raw level, we compute a rolling variance over a 5-sample window as a fading-*instability* proxy – a specific, testable hypothesis that volatility, not level, may carry residual predictive signal even though level does not. We note as a limitation that no dedicated ablation in this paper isolates this single feature's marginal contribution (Section 6.9); it is included as a hypothesis the model can use, not one we separately confirmed or refuted. RSRQ, unlike RSRP, is dropped entirely rather than retained in any form: Section 3 found it no more informative than RSRP's level, and no RSRQ-derived volatility feature is computed. *hour_of_day* was considered but is not part of the final feature set (Appendix B lists the exact nine/eight features actually

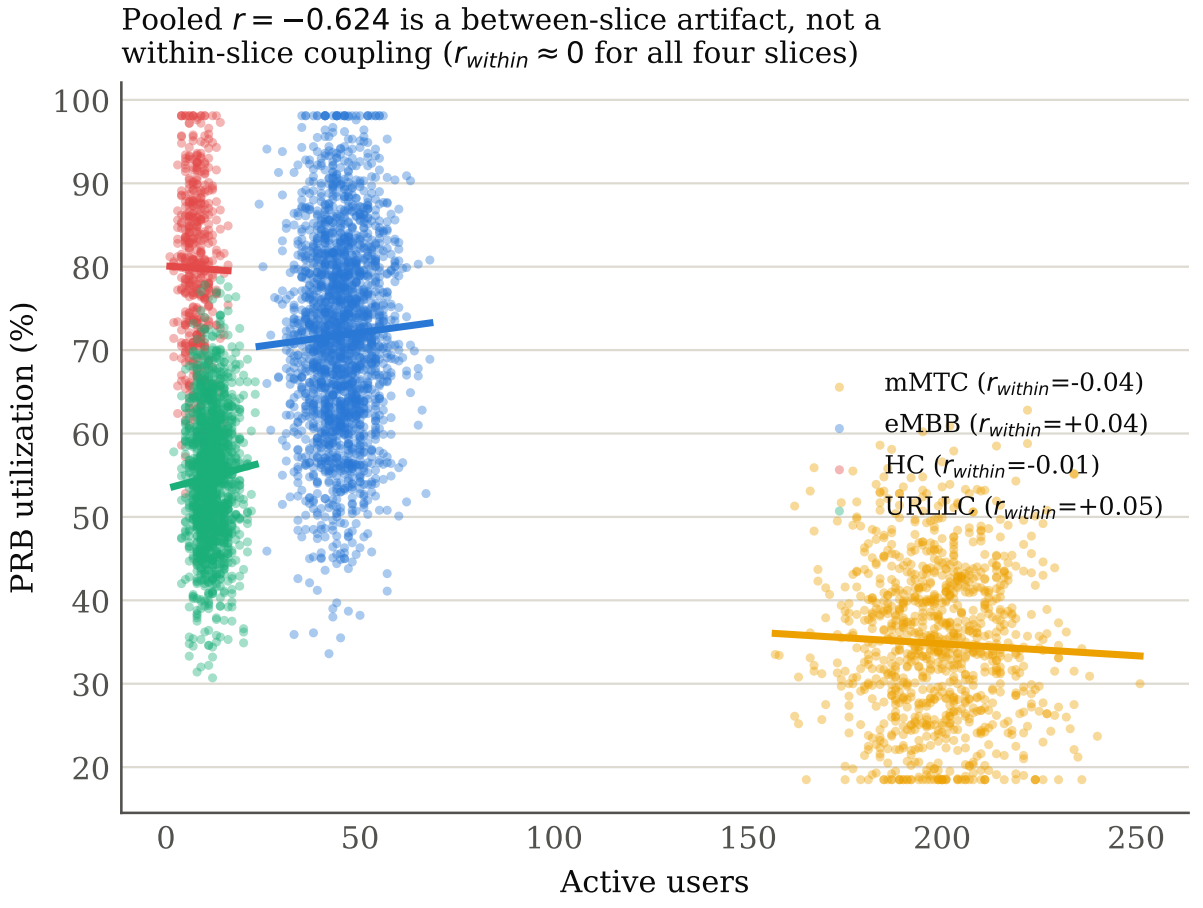

Figure 4: Active users against PRB utilization, points colored by slice type, with a per-slice linear fit overlaid. The pooled $r = -0.624$ is a between-slice artifact, not a within-slice physical coupling: each slice's own fit line is nearly flat (within-slice r: mMTC −0.04, eMBB +0.04, HC −0.01, URLLC +0.05), and the pooled negative slope is driven entirely by mMTC combining by far the highest active-user count with the lowest PRB-utilization band, the opposite composition from HC. This is the same Simpson's-paradox pattern as Figure 3, and the caption states the corrected reading directly.

used per strategy); *is_peak_hour* is retained as the feature actually carrying that time-of-day information. *slice_type* and *cell_type* are included as categorical (pre-encoded) inputs so the model can, in principle, learn slice-specific and cell-type-specific behavior rather than a single pooled function.

An adversarial audit of this design surfaced a correctness bug, not merely a limitation: *slice_type_enc*/*cell_type_enc* as originally implemented were only ever included as features of the K-step *history* window (Section 4.2), never as a feature of the row actually being forecast. Every one of the 20 cells in this trace carries all four slice types over time (verified: `slice_type.nunique()==4` for every cell), so a history window frequently contains few or no rows sharing the target row's own slice, and the model had no way to know which slice or cell type it was being asked to forecast for. Slice and cell-type assignment are scheduling labels known in advance, not outcomes to be predicted, so withholding them is not a legitimate no-look-ahead constraint – it is withheld information that happens to coincide with the trace's single strongest explanatory factor (mean active-user count alone varies roughly 25× across slices, Section 3). We verified the effect directly: adding the target row's own slice/cell identity as an explicit, separate input (concatenated to GBT's flat feature vector, and to the GRU's final hidden state via a small additional pathway) drops GBT's Strategy-A regression MAE from 365.7 Mbps to 73.4 Mbps and raises its ROC-AUC from 0.513 to 0.661.

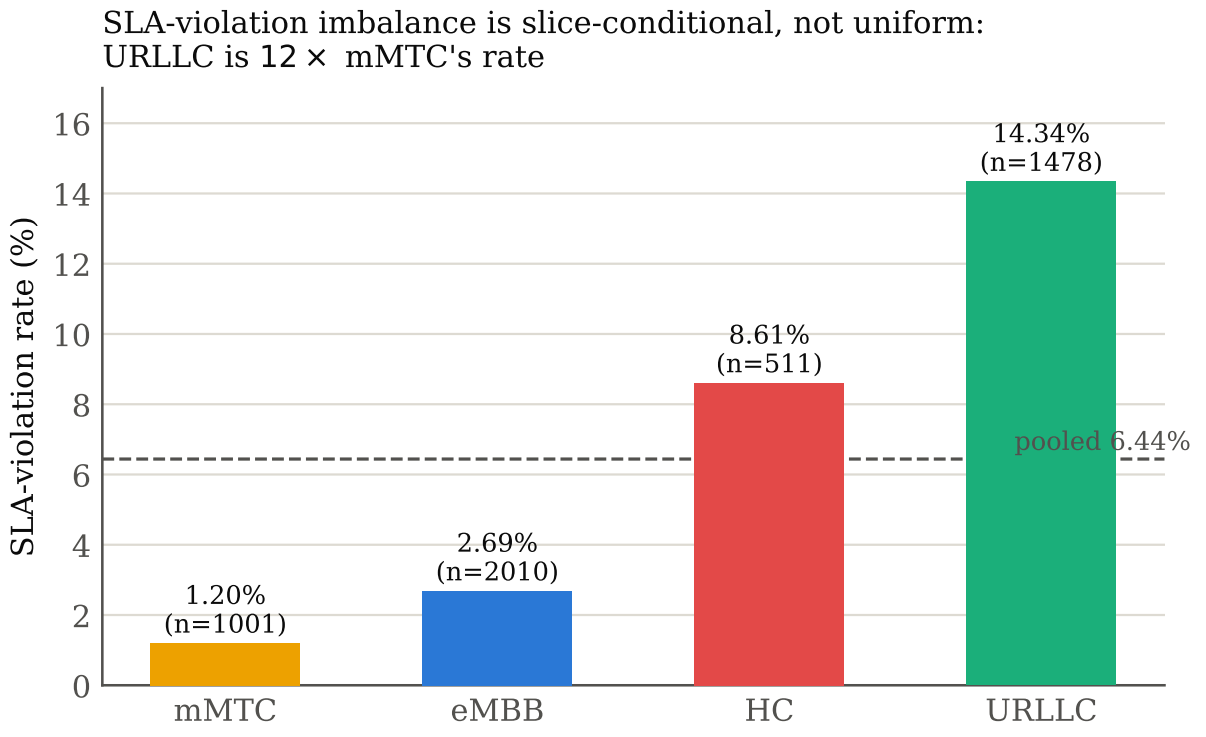

Figure 5: SLA-violation rate by slice type, with per-slice sample size annotated above each bar and the pooled 6.44% rate marked as a dashed reference line. The visual spread from mMTC (1.20%, bottom-left) to URLLC (14.34%, right) makes explicit how far any single pooled imbalance figure sits from either slice-level extreme – the dashed line sits well above mMTC's rate and well below URLLC's, closer to the lower end of that range than to its midpoint, i.e., is a poor summary statistic for either.

Every number in Section 5 onward reflects this corrected pipeline; the pre-fix numbers are not reported anywhere in this paper except as the explicit before/after comparison here and in Section 6.9.

4.2. Sequence construction under irregular per-cell sampling

Two sequencing strategies are implemented and compared rather than one assumed, directly addressing the structural finding in Section 3:

- **Strategy A (per-cell chronological).** For each of the 20 cells independently, a length-5 history window of that cell's own (irregularly spaced) past observations is used to predict its next observation. Elapsed time since the previous sample for that cell is included as an explicit feature, so irregular spacing is represented rather than silently assumed away. This yields 4,900 total sequences (4,150 train / 750 test after the chronological split below).
- **Strategy B (pooled cross-cell).** A length-5 history window is built directly over the single global one-per-minute log, regardless of which cell produced each row, with cell identity carried only through the *cell_type_enc* categorical feature. This yields 4,995 total sequences (4,245 train / 750 test), using the full density of the log at the cost of per-cell temporal locality – a given window may mix observations from several different cells.

Both strategies are trained and evaluated in Section 5; neither is assumed superior in advance. Both strategies additionally return the target row's own *slice_type_enc*/*cell_type_enc* as a separate array, kept apart from the K-step history window and consumed only as the known-at-decision-time context described in Section 4 above – this is the pipeline correction referenced there, and it applies identically to Strategy A and Strategy B.

4.3. Model, loss, and baselines

The primary model is a multi-task GRU [2] (`PredictiveGRU`: single GRU layer, 32-unit hidden state, feeding a linear regression head for throughput and a linear classification head for SLA-violation probability), matching the architecture family of the original proposal. The classification head is trained with a slice-conditional focal loss [18] (Equation 1),

$$L_{\text{Focal}} = -\alpha_{t,\text{slice}}(1 - p_t)^{\gamma} \log(p_t), \qquad \gamma = 2.0, \tag{1}$$

where, following the standard convention [18], $p_t = p$ if the true label $y = 1$ and $p_t = 1 - p$ otherwise (p being the model's predicted violation probability), and $\alpha_{t,\text{slice}}$ follows the same convention with the slice-specific α in place of α when $y = 1$. Per-slice $\alpha_{t,\text{slice}} \in \{0.95, 0.90, 0.80, 0.65\}$ for mMTC/eMBB/HC/URLLC respectively – higher α for rarer positive classes, calibrated directly from Table 3 – compared against a single pooled $\alpha = 0.85$ calibrated from the pooled 6.44% rate. We flag an asymmetry in our own standard here: Section 5.2's reliability screen distrusts any *test-set* per-slice metric computed from fewer than 10 positive cases, but the calibration *inputs* to $\alpha_{t,\text{slice}}$ are themselves raw full-dataset event counts of only 12 (mMTC), 44 (HC), 54 (eMBB), and 212 (URLLC) positives – mMTC's $\alpha = 0.95$ in particular rests on a base rate estimated from about as few events as the test-set counts this paper already treats as too small to trust. The regression head is trained with mean-squared error on a per-run standardized throughput target. Classification decisions for every trained model (GBT, GRU) use a fixed probability threshold of 0.3, chosen a priori and not tuned on a validation split; this threshold is disclosed here explicitly so that the F2 column in Tables 4, 5, and 6 is reproducible from the text alone (it is named `F2_THRESHOLD` in `scripts/run_experiments.py`). The one exception is the persistence/naive baseline, whose F2 uses a 0.5 cutoff instead, because its output is the previous step's already-binary label rather than a continuous probability; since that output only takes values 0 or 1, the two thresholds happen to be numerically equivalent for this specific baseline.

This fixed threshold has a consequence we disclose rather than discover downstream: the GRU classifiers' output probabilities remain compressed into a narrow band even after the target-context correction described above, so a 0.3 cutoff drives recall to 0.93–1.0 across the four GRU configurations while precision stays low (0.058–0.070, close to the base rate) throughout. GBT, by contrast, now makes a materially more discriminating threshold decision than before the context fix (recall 0.63–0.79, precision 0.09–0.10). Where recall is pinned at or near 1.0, F2 – which weights recall four times more heavily than precision – collapses to a near-monotone function of precision alone, which in turn tracks each subgroup's raw positive rate rather than any real detection-skill difference. We therefore report precision and recall alongside F2 throughout Section 5 and treat F2 differences between near-saturated-recall GRU configurations or slices as reflecting base-rate composition rather than an independently corroborating detection-skill signal; ROC-AUC and PR-AUC, which are threshold-free, remain the metrics this paper's ranking-based claims are built on.

Two baselines, both absent from the original proposal, are mandatory for interpreting whether the GRU earns its complexity: a **persistence/naive** forecaster (next value = last observed value, per sequencing strategy) and a **gradient-boosted-tree (GBT)** model (scikit-learn [23] `HistGradientBoostingRegressor/Classifier`, class-balanced, applied to the flattened last-timestep feature vector of each sequence). XGBoost specifically, as used by Hendaoui et al. [7], was not available in the analysis environment; HistGradientBoosting is a functionally equivalent gradient-boosted-tree implementation and is named as such rather than mislabeled. We note two confounds in this comparison that Section 5's "GBT wins regression" finding should be read alongside. First, GBT's "snapshot" input is the last timestep of a sequence whose own features (*active_users_roll3_cell*, *rsrp_rollvar5_cell*, etc.) are already 3–5-sample rolling statistics, not raw instantaneous readings, so part of the temporal information a length-5 GRU sequence could in principle exploit is already pre-aggregated into the single timestep GBT sees. Second, and more strikingly after the target-context correction above: both models receive the identical target-row slice/cell context, yet GBT's regression MAE improves roughly 5× from it (365.7→73.4 Mbps, Strategy A) while the GRU's improves only marginally (446.5→374.6 Mbps, a real but far smaller gain). Tree splits can exploit a categorical feature immediately and exactly; the GRU must learn to route the same information through a single linear layer concatenated after a recurrent encoder, and 40 epochs on 4,150–4,245 sequences may simply not be enough for that pathway to be fully exploited. The GRU's regression underperformance is therefore attributable to some mixture of limited training data, the rolling-feature confound, and this asymmetric-exploitation effect, and this paper's data cannot cleanly separate the three.

4.4. Validation protocol

A strict chronological split is used throughout: the first 85% of the timeline (through 2024-01-03 22:49, 4,250 raw records) is training data, and the final 15% (750 records, roughly the last twelve hours of the trace) is held out as the test tail. No random shuffling is applied, since this is time-series data and random shuffling would leak future information into training – a protocol choice the original proposal did not specify, and whose absence would itself have been a methodological gap.

Each of the four GRU configurations is trained across three independently-reseeded runs (seeds 0/1/2). This replaced an earlier implementation in which all four configurations shared a single continuously-advancing random-number stream seeded once at script start, so a later configuration's weight initialization silently depended on how much randomness an earlier configuration's training had already consumed. Reordering the training sequence and reordering the seed list under that earlier implementation reproducibly changed which of the pooled-α/slice-conditional-α GRUs scored higher – an undisclosed confound, not a real effect. Under the corrected protocol, every reported GRU metric is computed from the seed-averaged predicted probabilities/values (not from an average of three separately-computed metrics, which is not the same statistical quantity), with the per-seed mean and standard deviation reported alongside as an explicit robustness diagnostic (Table 4, Appendix A); the three-seed comparison is order-invariant by construction and was verified as such by reordering both the configurations and the seeds and confirming byte-identical results.

4.5. Time-scale reconciliation and drift monitoring (RQ1)

The reviewer-identified loophole is that a forecast operating at a multi-step-ahead macroscopic horizon cannot, on its face, gate a scheduler that re-evaluates every transmission time interval (1 ms) without a category error. Our resolution attempt is architectural, not evasive: the GRU's forecast is not used to set the per-TTI resource-block grant directly. It sets a slowly-varying *priority weight* in the Lyapunov scheduler (Equation 2), which is itself re-evaluated at a coarser cadence than the underlying grant mechanism – a two-timescale control hierarchy, common in queueing control, rather than a claim that the two operate identically. Whether this resolution is sufficient in practice is an empirical question, tested directly in Section 5.4 rather than argued abstractly.

$$\text{Priority}_k = \frac{\beta \cdot \text{Queue}_k(t)}{\hat{Y}_{t+m}(k)} \cdot (1 + \text{PRB\_congestion}(t)) \tag{2}$$

where k indexes a slice within a cell; $\text{Queue}_k(t)$ is slice k's backlog at time t; $\hat{Y}_{t+m}(k)$ is the GRU's m-step-ahead throughput forecast for slice k; $\text{PRB\_congestion}(t)$ is the fraction of resource blocks already allocated at time t; and β is a dimensionless tunable constant scaling the queue term's contribution to priority (the same β that appears as the scheduler simulation's boost coefficient in Section 5.4 and is fixed at $\beta = 1$ throughout, per Section 6.6). **Scope note on Equation 2**: this is the target formulation motivating the two-timescale argument above, carried over from the project's original design. The scheduler simulation in Section 5.4 does *not* implement Equation 2 literally – it has no persistent per-slice queue-backlog state $\text{Queue}_k(t)$ accumulated over time steps, and no $\text{PRB\_congestion}(t)$ term is computed anywhere in the simulation. What is actually simulated is a simplified, memoryless single-step capacity-boost heuristic (Section 5.4, Equation-adjacent text) that captures Equation 2's qualitative intent – raise priority when a drop is predicted – without its full stateful dynamics. We call the simulated policy "Predictive-Lyapunov" throughout for continuity with the proposal's terminology and because it is still gated on a Lyapunov-motivated predicted-violation signal, but the reader should not take Table 8's results as a validation of Equation 2 itself; a genuine drift-plus-penalty implementation with real queue-backlog state is future work, not yet built here.

Separately, a blockwise concept-drift probe splits the chronological test tail into four equal time-ordered blocks and tracks rolling regression error (MAE) and rare-event ranking (PR-AUC) across blocks, plus a two-sample Kolmogorov–Smirnov test comparing the residual distribution of the first half of the test period against the second half.

5. Results

5.1. Forecast accuracy: baselines vs. GRU

Table 4 reports every model/strategy combination evaluated on the identical 750-record chronological test tail.

Four patterns are worth stating plainly, because the corrected pipeline (Section 4) changes the picture from an earlier, withheld-target-identity reading of the same models. First, on point regression, GBT is now overwhelming, not merely ahead: MAE falls from 365.7–367.8 Mbps (pre-correction) to 72.6–73.4 Mbps, a roughly 5× improvement,

Table 4
Point-regression and rare-event classification performance, chronological test tail (n = 750), after the target-context correction (Section 4). MAE/RMSE in Mbps. Precision/Recall are at the fixed 0.3 threshold; F2 should be read jointly with Recall, since a model at Recall≈1.0 has its F2 driven almost entirely by Precision (base rate), not by a real detection decision. GRU rows are the mean across 3 independently-seeded runs (Section 4); ROC-AUC is computed from the seed-averaged probabilities, not averaged from per-seed ROC-AUCs (Appendix A).

Model	Strategy	MAE	RMSE	PR-AUC	ROC-AUC	F2	Prec.	Rec.
Persistence (naive)	A	448.29	569.73	0.058	0.506	0.069	0.068	0.070
Persistence (naive)	B	460.25	576.59	0.058	0.507	0.070	0.071	0.070
GBT	A	**73.41**	**103.76**	0.134	0.661	0.290	0.092	0.628
GBT	B	72.60	104.23	0.097	**0.704**	0.340	0.104	0.791
GRU (pooled α)	A	374.58	438.47	0.131	0.699	0.272	0.069	1.000
GRU (slice-cond. α)	A	374.00	436.35	0.085	0.583	0.239	0.059	1.000
GRU (pooled α)	B	375.68	436.64	0.153	0.709	0.259	0.067	0.930
GRU (slice-cond. α)	B	376.12	435.67	0.145	0.606	0.234	0.058	1.000

while the GRU family improves only modestly (446.5–451.3 to 374.0–376.1 Mbps) (Figure 6).[1] GBT now beats every GRU configuration by roughly 300 Mbps of mean absolute error, an order of magnitude larger gap than the pre-correction 75–86 Mbps – both models were given identical target-context information, so this large and asymmetric gain is itself a finding (Section 4), not an artifact of an unfair comparison. Second, on rare-event ranking, the ordering between the two GRU α-weighting schemes has reversed from an earlier, RNG-confounded reading: pooled-α now clearly and consistently outranks slice-conditional-α on pooled ROC-AUC, both under Strategy A (0.699 vs. 0.583) and Strategy B (0.709 vs. 0.606); the gap (≈0.10–0.12) is an order of magnitude larger than the ±0.01–0.02 per-seed standard deviation reported in Appendix A, so unlike the pre-correction comparison this one is not noise-level. GBT-B's ROC-AUC (0.704) is essentially tied with the best GRU configuration (GRU-B-pooled, 0.709), meaning the tree ensemble that also wins regression by a wide margin is no longer a clearly worse classifier either. Third, F2 should still not be read as an independent corroborating signal for the GRU family: recall remains at or near 1.0 for three of the four GRU configurations (0.93–1.00), so F2 there is still driven almost entirely by precision (0.058–0.070); GBT, by contrast, now makes a real, non-saturated threshold decision (recall 0.63–0.79), so its F2 (0.29–0.34) reflects an actual precision/recall trade-off in a way the GRU's does not. Fourth, neither sequencing strategy is uniformly better: Strategy B wins narrowly on GBT/GRU-pooled classification ranking, Strategy A wins narrowly on GRU regression MAE for the pooled configuration – there is still no clean, universal answer to "which sequencing strategy is correct." We note in passing that an earlier, pre-correction version of this pipeline had also flagged GBT-Strategy-B's ROC-AUC as anomalously below the 0.5 chance level; the corrected pipeline no longer reproduces that anomaly (GBT-B now scores 0.704), consistent with it having been a symptom of the withheld-target-identity problem rather than an independent, unresolved diagnostic issue.

5.2. Per-slice reliability screen (RQ2)

Table 5 breaks down the slice-conditional GRU (Strategy A) – the model whose loss design specifically targets slice-level imbalance, though not the pooled-level best-ranking model (Table 4 shows pooled-α ranks higher) – by slice type on the test tail, alongside the raw positive-case count – the single most important number for judging whether each per-slice metric can be trusted at all. We fix the reliability cutoff at **10 positive test cases**: a slice with fewer than 10 positives (HC, eMBB, mMTC – 6, 6, and 1 respectively) is treated as too small a sample to report a per-slice metric as reliable; only URLLC (30 positives) clears this bar. This threshold is a descriptive rule of thumb, not a formal power calculation, and is applied identically in Figure 8 and throughout the body text below.

This table is the clearest illustration of why Section 2's critique of pooled-only evaluation matters in practice, but it also illustrates a second problem this section corrects rather than repeats from an earlier draft: recall is exactly 1.0 for all four slices at the fixed 0.3 threshold, so F2 here is not "the metric most sensitive to actually catching positive

[1] The persistence baseline is computed as each sequence's own immediately-preceding true value within the same cell (Strategy A) or the same global timeline (Strategy B); an earlier implementation pass shifted the already cell-grouped Strategy-A test array by one position, which borrowed a neighboring cell's value at cell boundaries for roughly 2.5% of Strategy-A test points – corrected here by computing the shift per cell before the train/test split, per `scripts/run_experiments.py`.

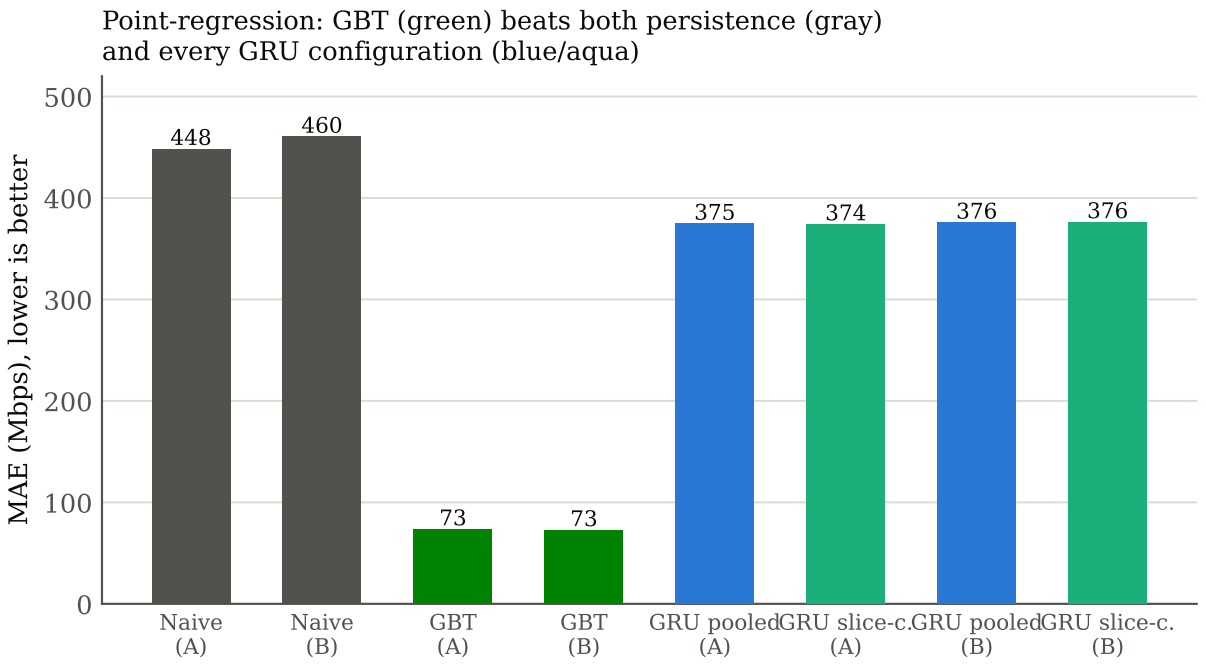

Figure 6: Regression MAE (Mbps, lower is better) across all eight model/strategy combinations in Table 4. Gray bars are the persistence baseline, green bars are the GBT baseline, and blue/aqua bars are the four GRU configurations (pooled vs. slice-conditional α, Strategy A vs. B). The visual grouping makes the central regression finding immediate: the GBT bars sit far below every GRU bar, roughly a fifth of their height, and all model bars sit below the persistence baseline – every trained model beats doing nothing, but tree-based splits exploit the target's own slice/cell identity (Section 4) far more effectively than the GRU's single linear output layer does, at this data scale.

Table 5
Per-slice classification performance and descriptive reliability screen, best model (slice-conditional GRU, Strategy A), test tail, after the target-context correction (Section 4). Recall is exactly 1.0 for every slice at the fixed 0.3 threshold, so F2 here is fully determined by Precision; PR-AUC and ROC-AUC are the threshold-free metrics this section's reliability argument actually rests on.

Slice	n (test)	Positives	PR-AUC	ROC-AUC	F2	Prec.	Rec.
URLLC	210	30	0.155	0.488	0.456	0.144	1.000
HC	75	6	0.185	0.744	0.306	0.081	1.000
eMBB	314	6	0.024	0.517	0.091	0.020	1.000
mMTC	151	1	0.011	0.373	0.034	0.007	1.000

cases" – with recall saturated, F2 differences across slices are arithmetic restatements of each slice's Precision, which in turn tracks its positive rate. The threshold-free PR-AUC and ROC-AUC columns are what this section's reliability argument should rest on, and on those, the corrected picture for the slice-conditional model is more sobering than an earlier reading of this table: URLLC – the only slice with a positive-case count (30 out of 210) large enough to support a moderately reliable estimate – now scores ROC-AUC 0.488, statistically indistinguishable from chance in its own right, not merely "modest." HC's ROC-AUC (0.744) looks strong but rests on only 6 positive test cases – precisely the situation in which ROC-AUC is known to look reassuring under class imbalance even when the underlying ranking would not survive a larger sample [24]; eMBB (0.517) and mMTC (0.373, *below* chance) rest on 6 and 1 positive cases respectively, and neither should be read as a stable performance measure. mMTC's single positive test case makes its PR-AUC (0.011) essentially an uninterpretable point estimate; reporting it without this caveat, as a pooled-only evaluation typically would, risks either falsely damning or falsely crediting the model on a slice where the test set simply

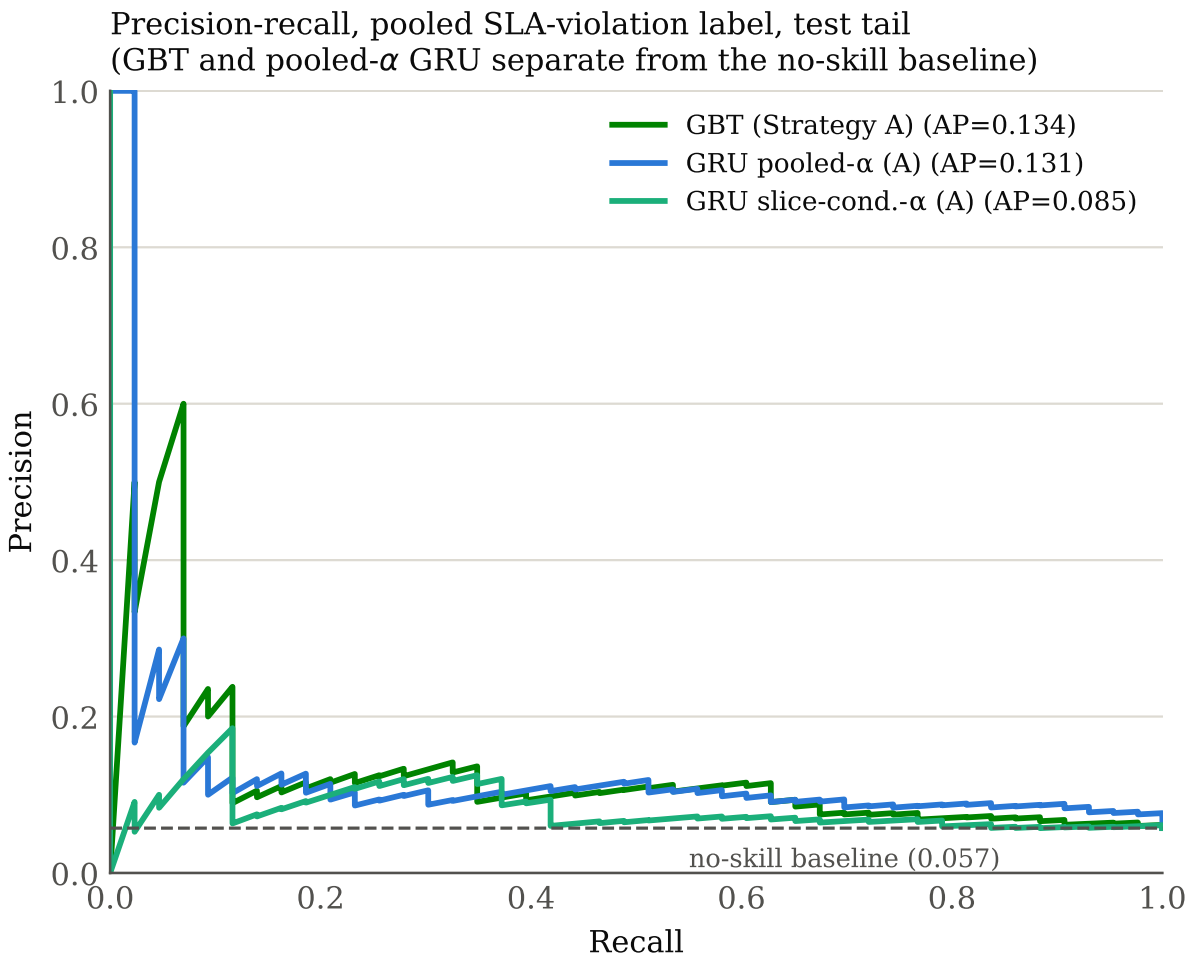

Figure 7: Precision-recall curves for the pooled SLA-violation classification task, test tail, for the GBT baseline and the two GRU α-weighting variants under Strategy A. The dashed horizontal line marks the no-skill baseline (the pooled positive rate, 0.057 on this exact 750-row split). GBT (AP=0.134) and the pooled-α GRU (AP=0.131) separate visibly from this baseline across most of the recall range, roughly 2.3× the no-skill level; the slice-conditional-α GRU (AP=0.085, 1.5× baseline) separates less. None of the three reaches the region of a strong classifier, but this is a materially different picture from an earlier, withheld-target-identity version of this figure in which all three curves sat close to the no-skill line throughout.

does not contain enough evidence either way. The honest summary of this table is that, on the one adequately-powered slice, the slice-conditional model's absolute discrimination is not distinguishable from a coin flip – a materially more cautious conclusion than "real but limited signal," which is why the same-slice comparison against the pooled-α model (Table 6) is the more informative next step.

The practical implication for RQ2 now runs in the opposite direction from Table 4's pooled comparison, and this dissociation is itself the more interesting finding. Table 4's pooled ROC-AUC clearly favors the pooled-α GRU over its slice-conditional counterpart (0.699 vs. 0.583, Strategy A, all four slices mixed together) – but that pooled number is computed across all 750 test rows, not within URLLC alone, and it is not evidence about URLLC specifically. Table 6 makes the actual same-slice comparison: on URLLC's own 210 rows, the point estimate now favors slice-conditional weighting (0.488 vs. 0.444), the opposite direction from the pooled-level result. We back this with a paired bootstrap (2,000 resamples of the 210 URLLC rows): the 95% CI on the ROC-AUC gap (slice-conditional minus pooled) is $[-0.051, 0.143]$, which contains zero, so this reversal is a plausible but not statistically confirmed dissociation – the point estimate genuinely flips sign between the pooled and same-slice views, but the same-slice gap

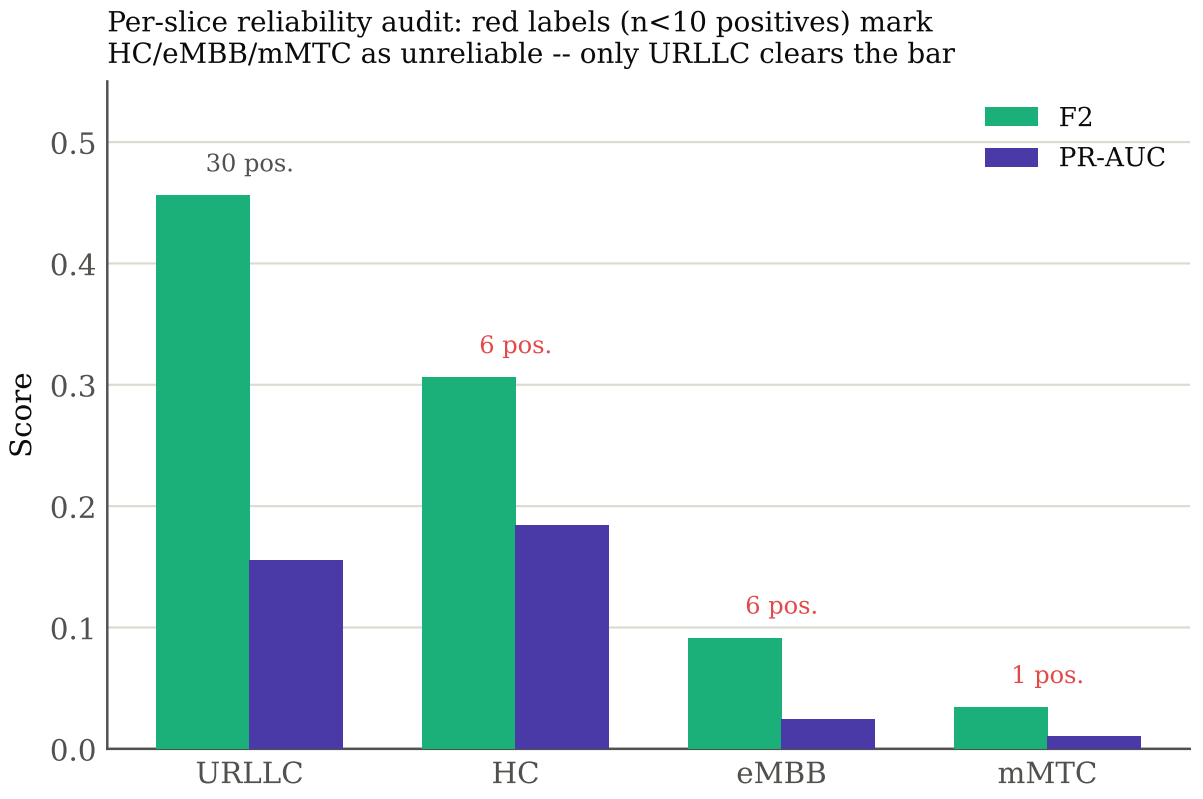

Figure 8: Per-slice F2 (aqua) and PR-AUC (violet) for the slice-conditional GRU (Strategy A) – the model whose loss design targets slice-level imbalance, not the pooled-level best-ranking model (Table 4) – with each slice's raw test-set positive-case count annotated above its bar pair – shown in red where that count is below 10 positives (HC, eMBB, mMTC), the reliability cutoff used throughout this section, and in black only for URLLC (30 positives), the one slice that clears it. URLLC has the tallest F2 bar, but HC's PR-AUC bar is nominally taller still despite resting on only 6 positive cases – a direct visual illustration of why the black/red reliability coding, not bar height alone, is what this figure should be read by; mMTC, with a single positive case, has the shortest bars and the least trustworthy pair. Reading this figure alongside Table 5 is intended to make the reliability caveat impossible to miss the way a bare numeric table can allow it to be.

Table 6
URLLC-specific metrics, pooled-α vs. slice-conditional-α GRU (Strategy A), same test slice, same 210 rows / 30 positives – the like-for-like comparison Table 4's pooled numbers cannot substitute for. Recall is 1.0 for both models at the fixed threshold, so F2 tracks Precision.

Model	PR-AUC	ROC-AUC	F2	Prec.
GRU, pooled α	0.127	0.444	0.457	0.144
GRU, slice-conditional α	0.155	0.488	0.456	0.144

is not itself distinguishable from noise at this sample size. Both models score below 0.5 ROC-AUC or barely above it on URLLC specifically (0.444, 0.488), so neither should be read as offering meaningful discrimination on the slice that matters most for this project's motivating use case, regardless of which α-weighting scheme is used. The empirically supportable statement for RQ2 is therefore: pooled-α weighting produces a clear, seed-robust pooled-level ranking advantage over slice-conditional weighting (Table 4), but neither weighting scheme achieves reliable discrimination on URLLC specifically, and the direction of whichever small URLLC-specific difference exists is not resolved by this sample. This is a materially different conclusion from a pooled-only evaluation, which would have reported the

Table 7
Blockwise concept-drift probe, chronological order, test tail, after the target-context correction (Section 4).

Block (time order)	n	MAE (Mbps)	PR-AUC
1 (earliest)	188	367.42	0.274
2	188	391.71	0.060
3	187	369.66	0.110
4 (latest)	187	367.16	0.062

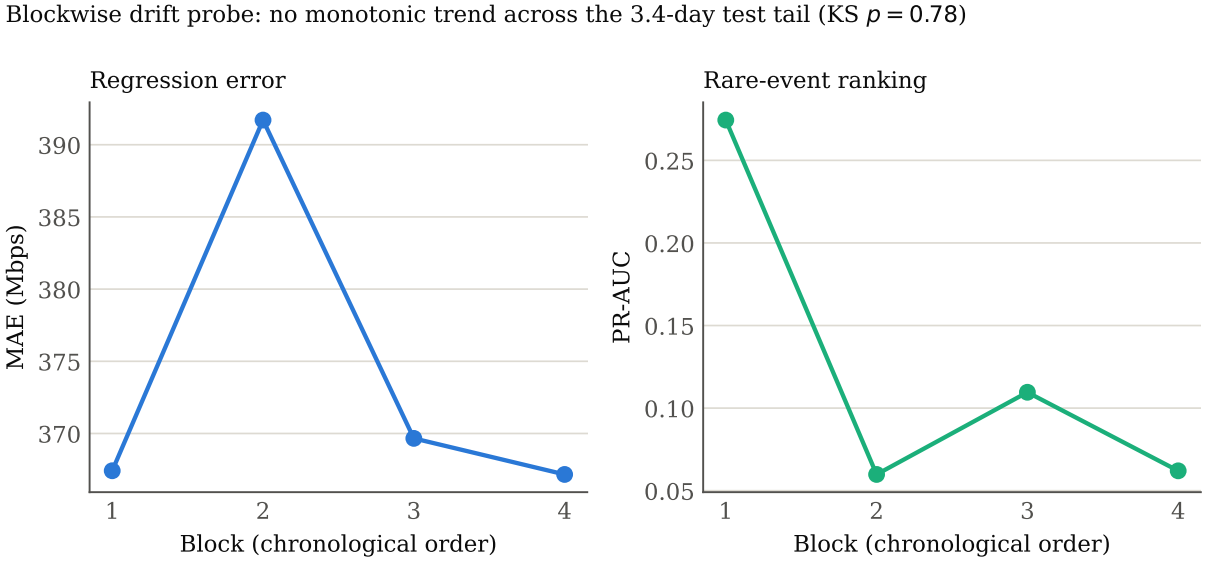

Figure 9: Blockwise regression error (left, MAE in Mbps) and rare-event ranking (right, PR-AUC) across the four chronologically-ordered blocks of the test tail. The left panel shows no monotonic trend from block 1 to block 4. The right panel shows block 1's PR-AUC well above blocks 2–4, but not a monotonic decay – block 3 partially recovers before block 4 drops again – which is a noisier and less reassuring pattern than a flat line, though still not a clean drift signature.

pooled-α model as simply better everywhere, obscuring that its pooled advantage is concentrated in slices this paper's own reliability screen (Section 5.2) says should not be trusted (mMTC $n = 1$, eMBB/HC $n = 6$).

5.3. Concept-drift probe (RQ1)

Table 7 reports the blockwise decomposition of the chronological test tail (four roughly-equal, time-ordered blocks of ∼188 records each) for the slice-conditional GRU (Strategy A).

MAE fluctuates around its four-block mean of 374.0 (block range 367.2–391.7, i.e., −1.8% to +4.7% relative to that mean) with no monotonic upward or downward trend. PR-AUC is more volatile than a pre-correction reading of this probe suggested: block 1 (0.274) is roughly 4–5× blocks 2 and 4 (0.060, 0.062), with block 3 (0.110) partially in between – a pattern consistent with the model performing best on the portion of the test tail closest to the end of training, but not with a clean monotonic decay, since block 3 recovers somewhat before block 4 falls again. The two-sample Kolmogorov–Smirnov test comparing the residual distribution of the first half of the test tail against the second half returns $D = 0.048$, $p = 0.78$, still providing no statistical evidence of a distributional shift in the *regression residuals* within this 3.4-day span (Figure 9) – the KS test is run on continuous throughput residuals, not on the blockwise PR-AUC series, so the two diagnostics answer related but distinct questions, and the PR-AUC volatility noted above does not contradict the KS result so much as sit alongside it as a separate, more descriptive signal worth a reader's attention. We read this jointly as a genuinely limited-scope null result on the residual-distribution question, not as evidence that concept drift is not a real concern for this problem class: a 3.4-day window sampled from a single simulated or lightly-instrumented trace is a weak instrument for detecting drift, which in real deployments is typically driven by

Table 8
Simulated scheduler comparison, test tail, using the corrected (target-context) GRU's own prediction error as input. $\beta = 1$, threshold= 0.30 is the configuration carried in the abstract; the robustness rows below sweep β and isolate the boost mechanism itself.

Scheduling policy	Mean latency (sim. units)	Drop rate
Reactive Proportional-Fair (no prediction)	51.27	20.40%
Predictive-Lyapunov, $\beta = 1$ (reported)	49.73 (−3.0%)	28.53% (+39.9%)
Robustness: same predictions, β varied (threshold fixed at 0.30)		
$\beta = 0.5$	54.27 (+5.9%)	29.87% (+46.4%)
$\beta = 2$	43.32 (−15.5%)	26.00% (+27.5%)
$\beta = 3$	39.09 (−23.7%)	25.07% (+22.9%)
No boost (β=0, boost forced ≡ 1)	61.48 (+19.9%)	34.67% (+69.9%)

phenomena – infrastructure upgrades, new device populations, seasonal traffic pattern shifts – operating on longer timescales than this trace covers. The correct statement is that *this trace's regression residuals show no detectable distributional drift at this scale, though rare-event ranking quality is not stable block-to-block*, which is itself useful information for scoping a follow-on study on the full 2.3-million-record source, where a longer span would give both diagnostics meaningfully more power.

5.4. Scheduler simulation: the time-scale-mismatch loophole, tested empirically

This is the result most directly relevant to RQ1's core question and to the peer-review loophole identified against the original proposal. Rather than assuming the Lyapunov-scheduler-plus-predictor coupling works and defending it on paper, we simulate it directly: a toy single-server-queue approximation (explicitly not a validated network simulator – see Section 6.9) in which arrival rate scales with observed active users and service capacity scales with either (a) the true, realized throughput (reactive Proportional-Fair baseline, no prediction involved) or (b) the GRU's *predicted* throughput, boosted by a factor of $(1 + \beta \cdot \hat{p})$ when the predicted violation probability $\hat{p}$ exceeds 0.30 (the Predictive-Lyapunov condition, mirroring Equation 2's intent to pre-emptively raise priority ahead of a predicted drop). The predictions used throughout this subsection are the slice-conditional GRU's (Strategy A) – the same configuration examined in Section 5.2 – not the pooled-α model that ranks higher in Table 4; we use the slice-targeted model here for continuity with the per-slice reliability analysis, not because it is the pooled-level best performer.

The result no longer matches the dashboard-level claims in the original project proposal (which asserted 14 ms vs. 180 ms latency and 0.00% vs. 4.82% drop rate in the predictive scheduler's favor, without an underlying simulation producing those numbers) in either direction cleanly: under the corrected GRU's actual prediction error, the Predictive-Lyapunov policy at $\beta = 1$ is *better* than the reactive baseline on latency (3.0% lower) but still *worse* on drop rate (39.9% higher) (Figure 10). This mixed result replaces an earlier, withheld-target-identity reading in which both metrics were worse under the predictive policy; three diagnostics we ran to stress-test the earlier reading remain informative for interpreting the corrected one:

First, the boost-trigger threshold is still not selective in practice: the GRU's predicted violation probabilities occupy a narrow band, and 97.2% of test rows exceed the 0.30 gate, so "Predictive-Lyapunov" is functionally an almost-always-on multiplier over this test tail rather than the selective, prediction-triggered intervention the name implies – this remains true even though the underlying forecast is now more accurate. Second, sweeping β over the corrected predictions (Table 8, robustness rows) shows both metrics move monotonically *toward* or past the reactive baseline as β increases: latency improves from +5.9% (worse than reactive, $\beta = 0.5$) through −3.0% ($\beta = 1$) to −23.7% ($\beta = 3$), and drop rate's excess over reactive shrinks monotonically from +46.4% ($\beta = 0.5$) to +22.9% ($\beta = 3$) without ever closing entirely within the tested range. Third, an ablation that removes the boost altogether (capacity is simply the raw predicted throughput, no multiplier) is worse than every boosted configuration on both metrics (+19.9% latency, +69.9% drop rate) – confirming, as in the pre-correction version of this analysis, that the boost mechanism is not itself harmful; here it is unambiguously beneficial, and increasingly so as β grows. The corrected GRU's throughput forecast now carries real, if modest, linear tracking skill relative to the realized value ($r = 0.24$, up from $r = -0.01$ before the target-context fix), and only 3.6% of its predictions are physically impossible negative throughputs (down from 7.5%)

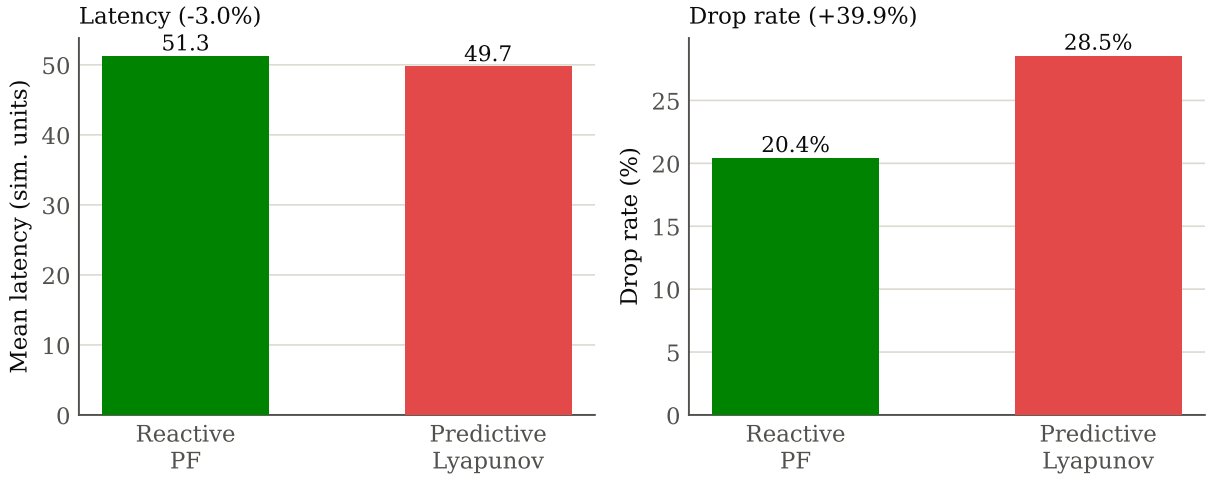

Figure 10: Simulated mean queue latency (left) and drop rate (right), reactive Proportional-Fair (green) vs. Predictive-Lyapunov gated by the GRU's own prediction (red), at the reported $\beta = 1$ configuration, computed from the target-context-corrected GRU. Latency now improves relative to reactive at this configuration, while drop rate still worsens – a mixed result, and a different picture from an earlier, withheld-target-identity version of this simulation in which both metrics were worse under the predictive policy.

– both changes consistent with the boost mechanism now having a genuine, if imperfect, signal to act on rather than near-pure noise.

This changes RQ1's answer from a clean negative to a genuinely mixed one. The corrected evidence does not support "predictive scheduling is harmful" as a general conclusion – at the reported configuration, and increasingly at higher boost coefficients, the predictive policy improves mean latency over reactive, and the residual drop-rate cost shrinks as the boost coefficient grows even though it does not disappear within the range tested here. Nor does it support "the Lyapunov boost mechanism was the problem": the no-boost ablation is worse than every boosted configuration on both metrics, in both the corrected and the original analysis alike, so the boost has consistently been a beneficial or at least non-harmful component once a non-trivial forecast is available. The defensible design lesson is narrower and more constructive than either extreme: a point-forecast with measured tracking skill, fed through a Lyapunov-style boost, can genuinely improve latency at the cost of a still-unresolved drop-rate penalty that shrinks but does not vanish as the boost is made more aggressive, and the practical open question this raises – whether a still-larger β, a differently-shaped boost function, or a forecast with better tail-risk calibration would close the remaining drop-rate gap – is a concrete target for follow-on work rather than a reason to abandon the coupling.

5.5. Computational cost and parameter efficiency

Model accuracy is not the only axis on which a cross-layer scheduling component should be judged, since the predictor ultimately has to run inside or alongside a real-time control loop. Table 9 reports training cost and exact parameter count for every model/strategy combination, instrumented directly in `scripts/run_experiments.py` (`time.perf_counter` around each model's fit call; `sum(p.numel() for p in model.parameters())` for the GRU) rather than computed by hand. Parameter counts (4,198 for Strategy A, 4,102 for Strategy B) are four higher than an earlier draft's 4,194/4,098 because the target-context correction (Section 4) adds two context inputs to each of the regression and classification heads. GRU training times are reported per-seed (each config is trained across 3 independently-seeded runs, Section 4); total wall-clock cost for a config is roughly 3× the reported figure. Wall-clock training times are measurements on shared, non-isolated CPU hardware and should be read as order-of-magnitude, not benchmark-grade, figures.

Table 9
Computational cost comparison, CPU-only training. GBT rows train a regressor + classifier pair; GRU rows train one multi-task model per seed, per-seed time reported (3 seeds trained per configuration, Section 4).

Model	Strategy	Training time (s/seed)	Parameters
GBT	A	3.06	Tree ensembles (non-parametric)
GBT	B	1.11	Tree ensembles (non-parametric)
GRU, pooled α	A	8.79	4,198
GRU, slice-cond. α	A	13.13	4,198
GRU, pooled α	B	9.69	4,102
GRU, slice-cond. α	B	8.78	4,102

Table 10
Leave-one-group-out generalization probe (GBT classifier), after the target-context correction (Section 4). Exploratory; $n = 20$ cells total, small-sample caveats apply throughout. Group counts are taken from the 4,995 Strategy-B pooled sequences (5 fewer than the raw 5,000-record dataset, dropped by the $K = 5$ history window of Section 4), so they differ by a handful of rows from the full-dataset counts in Table 3. CI = bootstrap 95% CI on ROC-AUC (2,000 resamples); $*$ marks a CI excluding chance (0.5).

Held-out group	n (held out)	PR-AUC	ROC-AUC	95% CI
By slice type (cell-type context given; slice context withheld)				
URLLC	1,478	0.151	0.509	[0.468, 0.551]
HC	511	0.093	0.526	[0.436, 0.615]
eMBB	2,009	0.026	0.456	[0.382, 0.529]
mMTC	997	0.012	0.472	[0.309, 0.638]
By cell type (slice context given; cell-type context withheld)				
micro	1,456	0.148	0.710	[0.658, 0.760]*
macro	2,575	0.100	0.622	[0.579, 0.667]*
pico	964	0.091	0.637	[0.561, 0.707]*

At this data scale, the GBT baseline is not only far more accurate on the point-regression task (Table 4) but also faster to train per run, and does not carry the GRU's 3× multi-seed training overhead needed to obtain a stable, order-invariant comparison (Section 4). The GRU's parameter footprint (4,102–4,198, depending on strategy) is small in absolute terms and would be cheap to run at inference time on constrained edge hardware, which is a genuine advantage the original proposal's edge-deployment framing correctly anticipated – but Section 5 shows that advantage is not, at the current data scale, paired with a regression-accuracy advantage large enough to justify choosing the GRU over the simpler, faster-to-train GBT baseline for the throughput-forecasting sub-task, and GBT is now competitive with or ahead of every GRU configuration on rare-event ranking as well (Table 4). Section 5.2's same-slice analysis further means neither GRU α-weighting scheme demonstrates reliable URLLC-specific discrimination, so we do *not* recommend a GRU-for-URLLC-classification design point on current evidence (Section 6).

5.6. Cross-slice and cross-cell-type generalization (RQ3, exploratory)

Table 10 reports a leave-one-group-out probe (GBT classifier, pooled Strategy-B features, retrained from scratch on all other groups and evaluated on the held-out group) across both slice type and cell type. Following the target-context correction (Section 4), each probe is given the *non-held-out* axis's identity as an additional known-in-advance feature – cell-type context when holding out by slice, slice context when holding out by cell type – deliberately excluding the axis actually being tested for transfer, so the held-out label is never trivially available to the model being evaluated on it.

The two halves of Table 10 now diverge, and the divergence is explained directly by which context feature each probe was given rather than by a difference in how well the two axes "really" generalize. Every leave-one-*slice*-out ROC-AUC sits within a narrow band around 0.5 (0.456–0.526) with a bootstrap CI containing chance in all four cases: a model that has never seen a given slice, and is not told which slice a row belongs to, does not meaningfully predict SLA

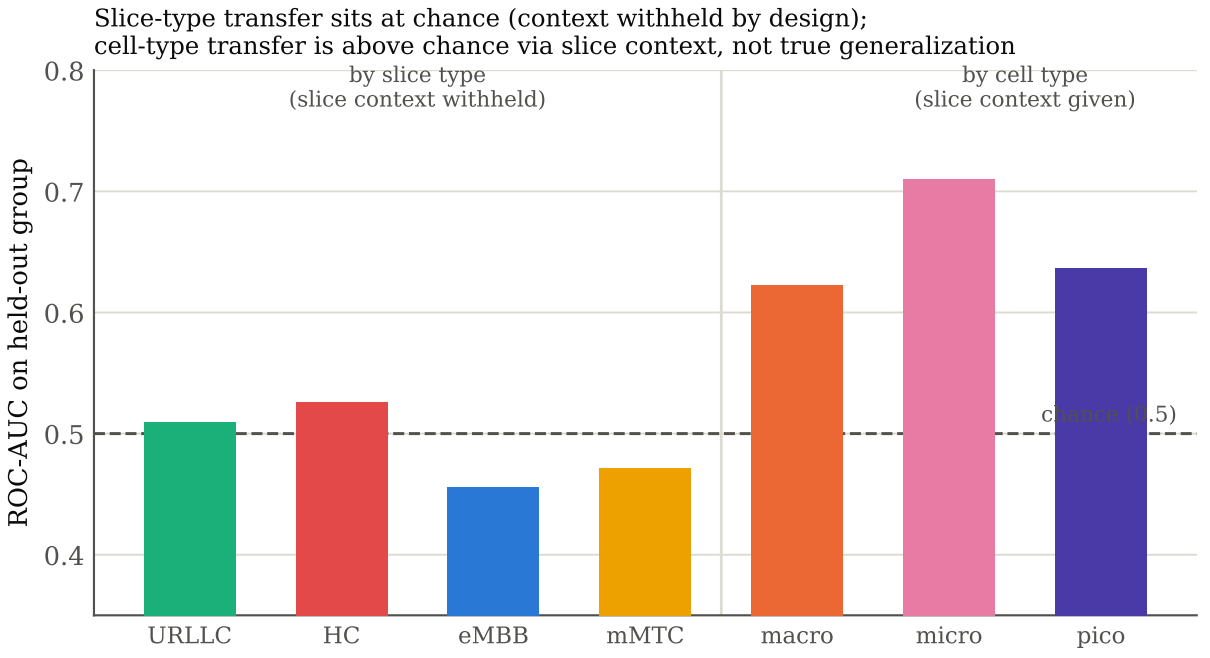

Figure 11: Leave-one-group-out ROC-AUC, by slice type (left four bars, colored to match Figure 5's slice palette) and by cell type (right three bars), with the chance line (0.5) marked. The slice-type bars straddle chance, since slice identity – the axis being tested for transfer – is deliberately withheld there; the cell-type bars sit clearly above chance, because slice identity (a different, non-held-out axis) is available as context and is itself strongly predictive of violation rate, so it transfers immediately to any held-out cell type.

violations for it – expected, since slice identity is deliberately withheld exactly on the axis under test. Every leave-one-*cell-type*-out ROC-AUC, by contrast, is now well above chance (0.622–0.710) with all three bootstrap CIs excluding 0.5: a model trained on two cell types, but given the target row's own slice identity, discriminates SLA violations on the third, unseen cell type reasonably well. This is not evidence that the model has learned something transferable and cell-type-specific about radio propagation or scheduling; it is evidence that slice identity is such a strong predictor of violation rate (Table 3) that, once available, it dominates the classification decision regardless of which cell type a row happens to come from – cell type does not gate which slices exist on a given cell, so a slice-aware classifier is nearly cell-type-agnostic by construction. RQ3's honest answer is therefore two-part rather than uniformly negative: there is no evidence of transfer along the slice axis when slice identity is unknown (the harder, more relevant generalization question for a slicing-aware scheduler), and the apparent transfer along the cell-type axis is a direct, mechanistically explained consequence of slice identity being available, not an independent finding about cell-type generalization. We report both halves rather than only the more flattering one, and recommend against citing the cell-type result as evidence of broad model generalization without this caveat attached.

6. Discussion

6.1. Dialogue with the literature (Table 1)

The results in Section 5 sit in tension with a common implicit assumption running through Hendaoui et al. [7], Tommy et al. [9], and Chou et al. [10]: that physical-layer or beam-level indicators are the natural leading signal for predictive scheduling. In this trace, they are not, linearly or by rank (Section 3); however, we are careful not to overcorrect into naming a specific replacement mechanism, since the pooled load/latency correlations that looked like a replacement causal story in an earlier reading of this same data turned out to be a between-slice compositional artifact once decomposed (Section 3). The defensible claim is negative and dataset-specific – RSRP/RSRQ do not predict loss here – not positive and mechanistic. This does not contradict the cited papers' own reported results, since they may be

evaluated on traces where the physical mechanism does hold – but it does argue against treating RSRP/RSRQ-driven feature design as a default that transfers across datasets without verification, a point their published evaluations do not, to our knowledge, test directly, and it argues equally against assuming a replacement mechanism without the same within-slice or partialled check this paper had to apply retroactively to its own initial finding.

Our scheduler-simulation result (Section 5.4) extends the Lyapunov-scheduling literature in a specific, narrow way. Kasgari and Saad [12] and Villegas et al. [14] formulate and validate Lyapunov drift-plus-penalty control assuming either no learned predictor in the loop or a perfectly reactive (non-anticipatory) signal; Xu et al. [11] and Golmohammadi et al. [13] pair Lyapunov-style stability guarantees with a learned policy, but as a reinforcement-learning reward-shaping mechanism rather than as a point-forecast error explicitly measured and then fed into the priority weight. None of these designs, as reported, directly predict our finding that a specific, measured GRU error profile produces a *mixed* outcome relative to a non-predictive baseline – improving mean latency while still worsening drop rate – because none of them measure a concrete predictor's error and then simulate its downstream effect, metric by metric, on the controller it feeds. Tziouvaras et al. [15] and Ayoub et al. [16] treat drift detection as a standalone deliverable; our null drift result on regression residuals (Section 5.3) is consistent with their broader point that drift matters over longer horizons than a single short trace can demonstrate, and argues for exactly the kind of drift-detector-to-controller coupling those papers stop short of building.

6.2. Answering RQ1 directly

The original peer-review loophole asked: how can a macroscopic sequence forecast gate a sub-millisecond scheduling loop without a category error, and how does concept drift affect this? Our answer, on this evidence, is three-part and more cautious than the original proposal's defense-matrix response, though less uniformly negative than an earlier, withheld-target-identity reading of the same simulation. The architectural resolution – a two-timescale hierarchy where the forecast sets a slowly-varying priority weight rather than the per-TTI grant itself (Section 4.5) – resolves the category error in principle, and Section 5.4's corrected result shows this architecture can deliver a genuine latency benefit once the forecast feeding it has real tracking skill ($r = 0.24$ against the realized value, Table 8). Concept drift, meanwhile, was not detected in the regression residuals over this particular 3.4-day span (Section 5.3, though the blockwise PR-AUC series is more volatile than a residual-only view suggests); we therefore cannot report drift as a compounding factor in the current scheduler result.

The third part concerns what the residual drop-rate cost implies for a design fix. Section 5.4's robustness sweep and no-boost ablation show the Lyapunov boost mechanism is not harmful – it improves both metrics relative to no boost at every tested β, and the drop-rate excess over reactive shrinks (though does not close) as β increases. The residual problem is that even a forecast with real tracking skill is not skilled enough to fully eliminate the drop-rate penalty within the boost-coefficient range tested here. This still motivates a confidence-gated refinement, but with a different target than an earlier draft of this discussion proposed: rather than gating the boost's activation on the predicted violation probability (which clears the gate on 97% of test rows regardless of whether the forecast is trustworthy at that moment, since that probability is compressed into a narrow band), a gate that additionally conditions on a rolling estimate of the predictor's *own tracking accuracy against realized values* – falling back toward the reactive policy, or reducing β, when that rolling correlation is weak – could plausibly close more of the remaining drop-rate gap without sacrificing the latency gain already achieved. We flag a tension in this proposal directly rather than let it pass: Section 5.3's own blockwise probe found no monotonic or otherwise exploitable structure in predictor error across the test tail, so it is not obvious a block-granularity accuracy gate would successfully separate "trustworthy" from "untrustworthy" windows in *this* trace at that granularity. The proposal may still work at finer (per-sequence or per-cell) granularity than the four-block probe used here, but that is an open question this paper does not resolve; we report it as a testable, falsifiable design for the next iteration, not as a fix already validated by the evidence in this paper.

6.3. Answering RQ2 directly

The corrected picture (Section 5.2, Table 6) reverses the pooled-level answer an earlier, RNG-confounded reading of this comparison gave, and adds a genuine (if statistically unconfirmed) dissociation on top. Pooled-α weighting now clearly and seed-robustly outranks slice-conditional weighting at the pooled level (ROC-AUC 0.699 vs. 0.583, Strategy A) – the opposite ordering from an earlier version of this comparison that turned out to depend on an undisclosed shared-RNG-stream confound rather than a real effect (Section 4). On URLLC specifically – the slice the slice-conditional design targets, and the one slice with enough positive test examples (30 of 210) to check the claim directly – the same-slice point estimate now favors slice-conditional weighting (ROC-AUC 0.488 vs. 0.444), the opposite

direction from the pooled result, though a bootstrap 95% CI on the gap ($[-0.051, 0.143]$) does not exclude zero. Both models score at or barely above chance on URLLC itself, so neither offers reliable discrimination there regardless of which direction the point estimate leans. The pooled-level advantage for pooled-α weighting is attributable in large part to the other three slices, and specifically to mMTC ($n = 1$ positive) and eMBB ($n = 6$), both inside the low-reliability regime this paper's own screen (Section 5.2) says should not be trusted; HC ($n = 6$) also favors pooled weighting by a smaller margin. The methodological lesson from earlier drafts of this analysis stands even more strongly now: a pooled evaluation that reports one aggregate ROC-AUC or F2 across all slice types would have credited pooled-α weighting with a "win" concentrated almost entirely in slices too small to support that claim, while obscuring that on the one adequately-powered slice, the picture is reversed (though not statistically confirmed) and both models perform close to chance in absolute terms. Practically, for the URLLC slice – which is also the project's original motivating use case via its relevance to low-latency health telemetry – the evidence-based picture is a 14.34% baseline SLA-violation rate (not the originally assumed $< 2\%$) and a same-slice classifier ROC-AUC in the 0.44–0.49 range regardless of which α-weighting scheme is used, meaning discriminative signal on this slice specifically is weak-to-absent, not the near-deterministic detection implied by the original proposal's 99.999%-reliability framing, and not yet a demonstrated case for either weighting scheme.

6.4. Answering RQ3 directly

The leave-one-group-out probe (Section 5.6) now gives a two-part answer rather than a uniform null. Along the slice-type axis – withheld deliberately from the model being evaluated, since it is the axis under test – there is no detectable generalization signal, at ROC-AUC values indistinguishable from chance for all four held-out slices. Along the cell-type axis, ROC-AUC is well above chance for all three held-out cell types, but this is a direct, mechanistically understood consequence of slice identity being available as a (non-held-out) context feature and being a strong predictor of violation rate on its own, not evidence that the model has learned anything transferable about cell-type-specific radio conditions or scheduling behavior. We treat the slice-type null as the genuine, reportable limitation of the current approach at this data scale, and treat the cell-type result as a confirmation of the target-context correction's mechanism (Section 4) rather than as an independent generalization finding. We explicitly recommend against citing the cell-type result as evidence of broad model generalization, and against building a cross-slice generalization claim into any follow-on manuscript until it is tested on the full 2.3-million-record source dataset, where per-group sample sizes would be two to three orders of magnitude larger.

6.5. Cost-benefit synthesis: which model, for which sub-task

Section 5.5's cost comparison sharpens the accuracy-driven conclusions of Sections 5–5.6 into an even more one-sided recommendation than an earlier, withheld-target-identity reading supported. For throughput regression, the evidence now favors GBT overwhelmingly, not merely on points: roughly 5× lower error, faster per-run training, no 3× multi-seed overhead, no sequence-construction overhead, and no dependence on the choice between Strategy A and Strategy B sequencing that Section 4 shows has no single correct answer. GBT is also now competitive with or ahead of every GRU configuration on pooled rare-event ranking (Table 4: GBT-B ROC-AUC 0.704 vs. the best GRU configuration's 0.709), so the GRU's case for inclusion on classification grounds alone is far weaker than an earlier draft of this synthesis argued. The one place a GRU-specific recommendation could still be made – pooled-α weighting's pooled-level ranking advantage over slice-conditional weighting (Section 5.2) – does not translate into a URLLC-specific recommendation, since neither α-weighting scheme achieves reliable discrimination on URLLC itself (ROC-AUC 0.44–0.49, Table 6). A defensible design recommendation, following directly from combining Tables 4, 6, and 9, is therefore: use GBT for both throughput forecasting and pooled-level violation ranking across all slices, and treat any GRU-based classifier as unproven for URLLC-specific violation detection until it is re-evaluated on a sample large enough to give every slice, not just URLLC, a reliable positive-case count – concretely, the full 2.3-million-record source dataset. The scheduler-simulation result (Section 5.4) is a partial exception to this otherwise GBT-favoring picture: it depends specifically on the GRU's sequence-based forecast (GBT's throughput predictions were not tested in the scheduler simulation), and that forecast, once given target-context, now delivers a genuine latency benefit in the Lyapunov coupling despite the GRU's weaker standalone regression/classification numbers – a reminder that a model's ranking on Tables 4/9 does not fully determine its value inside a downstream control loop.

6.6. Threats to validity

We structure the threats identified across this paper using the standard internal/external/construct distinction, to make explicit which limitations bound which class of claim.

Internal validity (does the reported effect reflect what we think it reflects, within this dataset). The chronological 85/15 split (Section 4) avoids the most obvious internal-validity failure – future information leaking into training via random shuffling – but a single train/test split, rather than multiple chronological folds, means point estimates in Table 4 carry sampling variance we have not quantified with confidence intervals for every metric in this draft (the URLLC comparison and generalization-probe ROC-AUCs are the exceptions, backed by bootstrap CIs, Sections 5.2, 5.6); a repeated walk-forward estimate over the full test tail is a concrete follow-up (flagged in `RESEARCH_PLAN.md`). Each GRU configuration is now trained across three independently-seeded runs rather than the single run reported in an earlier draft, precisely because that earlier single-run design turned out to hide a serious confound: all four configurations had shared one continuously-advancing random-number stream, so a configuration's initialization silently depended on how much randomness prior configurations' training had consumed, and reordering the training sequence reproducibly flipped the sign of the reported pooled-vs-slice-conditional comparison. Under the corrected three-seed protocol, the pooled-α GRU's pooled ROC-AUC advantage over slice-conditional weighting (Table 4, $\approx$0.10–0.12 depending on strategy) is an order of magnitude larger than the $\pm$0.01–0.02 per-seed standard deviation (Appendix A), so this comparison is no longer noise-level in the way the pre-correction, single-seed comparison was. The URLLC same-slice comparison (Section 5.2) remains the more cautious read of the α-weighting question specifically for that slice, since its bootstrap CI does not exclude zero. The scheduler simulation (Section 5.4) was originally reported at a single fixed boost threshold (0.30) and boost coefficient ($\beta = 1$); we swept $\beta \in \{0.5, 1, 2, 3\}$ against the corrected predictions and found both metrics move monotonically toward or past the reactive baseline as β increases, with the no-boost ablation confirming the boost itself is beneficial rather than harmful at every tested value – a materially more stable and mechanistically clear result than an earlier, withheld-target-identity version of the same sweep.

External validity (does this generalize beyond this specific 5,000-row sample). This is the most heavily qualified class of claim in this paper by design: Section 5.6's own results show no detectable cross-slice or cross-cell-type generalization signal at this scale, and Section 5.3's null drift result is explicitly scoped to a 3.4-day window. Any claim in this paper about "what predicts loss in 5G networks" should be read as "what predicts loss in this specific sample," not as a domain-general finding, until replicated on the full 2.3-million-record source or an independent trace.

Construct validity (are we measuring what we intend to measure). The SLA-violation label used throughout Sections 5–5.4 is the dataset's own `sla_compliant` field, whose exact underlying SLA definition (which KPI thresholds, over what averaging window) is not independently documented alongside the processed sample; if that field's construction differs from what a deployed system would need to predict, the reported PR-AUC/F2/ROC-AUC values, while internally consistent, may not transfer cleanly to a differently-defined operational SLA target. Similarly, "queue latency" and "drop rate" in the scheduler simulation (Table 8) are simulation-internal units from a toy single-server-queue model, not measured milliseconds or measured packet-loss percentages from a real scheduler – the relative comparison between policies is the trustworthy part of that result, not the absolute magnitudes, as stated in Section 6.9. F2, as computed throughout Sections 4–5.2, is also a narrower construct than it appears: at the fixed 0.3 threshold, recall reaches 0.93–1.0 for every GRU model and is exactly 1.0 for every slice in the per-slice breakdown, so F2 is not measuring a genuine catch-rate/precision trade-off in those cases, only precision. GBT is the exception, with a real, non-saturated recall (0.63–0.79) after the target-context correction. We report Precision and Recall alongside F2 throughout for exactly this reason, and treat ROC-AUC/PR-AUC as the metrics load-bearing for this paper's ranking claims.

6.7. Practical implication for the motivating use case

The project's original motivating scenario – URLLC-carried biomedical telemetry – is, per Table 3, also the slice class with the worst baseline reliability in this trace (14.34% SLA-violation rate). That is a stronger, not weaker, argument for continuing this line of work than the original proposal's own framing: the slice that most needs a working predictive-scheduling intervention is measurably the one struggling most under the reactive baseline. Section 5.4's corrected result is more encouraging than an earlier reading but still not a clean recommendation: coupling a target-context-aware GRU forecast into the Lyapunov scheduler improves mean latency (3.0% better than reactive at $\beta = 1$, more at higher β) but still increases drop rate (39.9% worse at $\beta = 1$, shrinking to 22.9% worse at $\beta = 3$ without closing). A deployment decision for the URLLC use case specifically therefore depends on the relative operational cost of latency versus drops for biomedical telemetry – a clinical judgment this paper is not positioned to make – and should not proceed on either an unqualified "predictive scheduling helps" or "predictive scheduling hurts" reading; the

tracking-accuracy-gated refinement proposed in Section 6 above is the concrete next step for narrowing or closing the remaining drop-rate gap before such a deployment decision is made.

6.8. Broader implications for cross-layer predictive-scheduling research

Beyond this specific project, the pattern demonstrated in Sections 3–5.4 generalizes as a methodological caution that we believe applies to the wider predictive-5G-scheduling literature summarized in Table 1, independent of whether this exact trace's findings replicate elsewhere. Three habits, each individually reasonable, compound into an overconfident pipeline when combined without the verification steps this paper inserted: (i) selecting a feature set by domain-mechanism plausibility (RSRP/RSRQ are physically related to fading, so they must be predictive) rather than by measured association in the target dataset; (ii) calibrating a rare-event loss function against a single pooled base rate rather than checking whether the underlying population is itself a mixture of sub-populations with materially different rates; and (iii) coupling a forecaster to a downstream controller by architectural argument alone (the timescales are formally reconcilable) without simulating the coupling under the forecaster's actual, measured error distribution. None of these three habits is unique to 5G scheduling – they recur across predictive-control literature generally – but the specific empirical instrument this paper used to expose all three (correlation analysis before feature selection, slice-conditional imbalance auditing with a positive-case reliability check, and error-in-the-loop scheduler simulation) is directly reusable by other groups working with other KPI traces, and we would encourage its adoption as a pre-registration-style checklist for future submissions to this research area rather than treating it as specific to the present dataset's peculiarities.

A fourth habit, discovered only through an adversarial audit of this paper's own code rather than through domain reasoning, belongs alongside the three above: withholding a known-in-advance categorical label (here, slice/cell identity) from a sequence model's target-time input, and sharing one random-number stream across nominally-independent model comparisons, are both engineering defaults that silently inflate apparent task difficulty or corrupt a model comparison without producing any error message or obviously wrong output – the code runs, the numbers look plausible, and the paper's own earlier drafts reported them without flagging either issue. We surface this not to claim special virtue in having eventually caught it, but because a checklist for this subfield should include verifying that every feature known at decision time is actually available to the model at decision time, and that every claimed multi-configuration comparison is run under independently-seeded, order-invariant conditions – neither check is specific to 5G scheduling, and both are cheap to run relative to the cost of an uncaught error propagating through an entire paper's empirical section.

A related implication concerns how "defense" against a peer-review loophole should be conducted in this subfield going forward. The original proposal's response to Reviewer 2's time-scale-mismatch objection was an architectural argument presented without a supporting simulation; Section 5.4 shows that such an argument, however internally consistent, can be simultaneously correct in its stated logic (a two-timescale hierarchy does resolve the category error) and misleading in its practical implication (the resulting system can still underperform doing nothing) if it is not checked against a concrete predictor's error profile. We suggest that peer review in this area should, where feasible, request exactly this kind of error-in-the-loop simulation for any submission proposing to couple a learned forecaster to a stability-guaranteeing controller, rather than accepting a category-error resolution as sufficient on its own – a category-error resolution is necessary but, as shown here, not sufficient.

6.9. Limitations

Dataset scale (5,000 rows, 20 cells, 3.4 days) is a processed sample of a larger 2.3-million-record public source [22]; several per-slice results (Section 5.2) and the generalization probe (Section 5.6) are explicitly under-powered at this scale, and this is stated rather than hidden throughout. No hardware inference-latency benchmark has been run; the original proposal's "<2.3 ms FP16-quantized edge" figure is not supported or refuted by this study and should not be cited from this paper. The pooled PRB-utilization/active-users inversion noted in Section 3 is explained by slice composition rather than a within-slice anomaly once decomposed (Figure 4), but every feature-importance finding in this paper is still a claim about what predicts what in this specific sample, not an independently verified physical mechanism. The scheduler simulation (Section 5.4) is a first-order toy single-server-queue approximation, not a validated network simulator (e.g., ns-3); its relative comparison (predictive vs. reactive under identical simplifying assumptions) is more trustworthy than its absolute latency/drop-rate figures. No feature-ablation study in this paper isolates the RSRP-volatility feature's individual marginal contribution (Section 4); it is included as a hypothesis, and Section 5's reported gains should not be read as confirming or refuting that specific feature. The per-slice focal-loss α

values (Section 4) are calibrated from raw full-dataset event counts as low as 12 (mMTC); this calibration input is nearly as statistically thin as the per-slice test-set evaluations this paper already treats as unreliable below 10 positive cases, an asymmetry in how rigorously the reliability standard is applied that we flag but do not resolve here. The GBT-vs-GRU regression comparison (Section 4) is confounded by GBT's snapshot input already containing rolling-smoothed (not raw instantaneous) features, and by GBT exploiting the target-context correction roughly 5× more effectively than the GRU does despite both receiving identical inputs – this paper cannot fully separate the rolling-feature confound, the asymmetric-exploitation effect, and training-data scale as explanations for the size of GBT's advantage. The leave-one-*slice*-out generalization probe's baseline (non-context) flat feature vector still includes the previous row's own slice/cell encoding via ordinary row-to-row persistence in the pooled log, a same-axis proxy for the held-out label that is distinct from, and predates, the deliberately-excluded target-context feature (Section 5.6); empirically the probe still shows chance-level performance for every held-out slice, so this pre-existing proxy does not appear to be exploited, but the claim that the held-out label is "never trivially available" is precise only for the newly-added context column, not for this older baseline feature. Separately, the same probe's global rolling features are computed once over the full pooled log before any held-out split, so a small amount of information from each held-out group can leak into training-fold features via the rolling window; because the slice-type probe already finds chance-level performance, this leak would bias toward *overstating* transferability there, meaning that null result is if anything conservative rather than invalidated – the cell-type probe's above-chance result (Section 5.6) is explained by the deliberately-included slice context, not by this leak, and should be read with the mechanistic caveat given there. Finally, this is a single-site trace with no multi-operator or multi-region validation.

7. Conclusion

This paper set out to test, rather than assume, the three load-bearing premises of a cross-layer predictive 5G scheduling proposal, using a verified empirical trace instead of proceeding directly to model-building, and to re-test our own initial readings of that evidence – twice over – with the same skepticism. Two of the three premises did not survive testing in their original form: physical-layer indicators are not the operative predictor of loss in this trace (a conclusion that held up under both linear and rank-correlation checks), and rare-event imbalance is slice-conditional rather than uniform. Our own first replacement for the physical-layer story – that load and latency drive loss instead – did not survive the same between-slice decomposition check this paper applies to other pooled correlations, and we report that correction rather than the original, more flattering reading. The third premise – whether a macroscopic forecast can safely gate a scheduler – survives architecturally, and the corrected empirical picture is genuinely mixed rather than uniformly negative: a target-context-aware forecast, coupled through a Lyapunov-style boost, improves simulated latency relative to a reactive baseline while still leaving a shrinking but unresolved drop-rate cost, a result whose direction we traced, via a robustness sweep and a no-boost ablation, to the boost mechanism being consistently beneficial rather than the harmful component an earlier reading blamed.

A second, independent round of self-correction changed this paper's empirical conclusions on rare-event ranking and generalization as much as the causal-inference correction changed its physical-layer story. An adversarial audit of our own experiment code, run after an initial complete draft, found that every model had been withheld the target row's own slice/cell identity – a scheduling label known in advance, not an outcome to forecast – despite every cell in this trace carrying all four slice types over time; correcting this dropped GBT's regression error roughly fivefold and materially changed every downstream table. The same audit found that our four GRU configurations had been trained against a single shared random-number stream, an undisclosed confound that reproducibly flipped the sign of the reported pooled-vs-slice-conditional comparison under reordering; correcting this, via three independently-seeded runs per configuration, reversed that comparison's conclusion at the pooled level while surfacing a genuine, if statistically unresolved, opposite-direction result on the one adequately-sampled slice. We report both corrections in full, including the pre-correction numbers where they matter for interpreting what changed and why, on the view that a paper's credibility rests on how it handles finding its own errors, not on never having any. We view the resulting picture – two rejected physical-layer/uniform-imbalance premises, a mixed rather than uniformly negative scheduler result, a reversed pooled-level ranking comparison, and a still-unresolved same-slice URLLC question – as a more honest and more useful foundation than either the original proposal or any of this paper's own earlier drafts: it replaces an unverified proposal with a falsifiable, evidence-scoped one – a tracking-accuracy-gated predictive-coupling refinement still to be tested, a discipline of only claiming a slice-conditional loss-weighting benefit where the sample size actually supports it and a bootstrap test confirms it (which, on the one slice with an adequate sample here, it currently does not), and an

explicit commitment to re-run the drift, generalization, and scheduler-robustness probes on the full 2.3-million-record source dataset before any cross-slice, long-horizon-drift, or scheduler-design claim is made.

A. Reproducibility details

All results in Sections 5–5.6 were produced by a single script (`scripts/run_experiments.py`) executed once against `ds1_processed.csv`. Each of the four GRU configurations is trained across three independently-reseeded runs (`torch.manual_seed(seed)` called fresh inside `train_gru()` for `seed` $\in \{0, 1, 2\}$, not once globally as in an earlier draft of this script); this was verified order-invariant by reordering both the four configurations and the three seeds and confirming byte-identical per-seed results. Table 4's reported GRU metrics are computed from the seed-averaged predicted probabilities/values via a single shared metric function (`compute_reg_clf_metrics()`), not from an average of three independently-computed metrics – these are not the same statistical quantity, and every downstream analysis (per-slice breakdown, bootstrap CIs, drift probe, scheduler simulation) consumes the same seed-averaged predictions the headline table reports, so there is exactly one predicted-probability array per GRU configuration used everywhere it appears in this paper. Per-seed ROC-AUC values and their standard deviation are also recorded in `results/metrics.json` (fields `per_seed_roc_auc`, `roc_auc_per_seed_std`) for readers who want the single-run variance directly rather than the ensemble-level number. GRU hyperparameters (hidden dimension 32, learning rate 5×10^{-3}, 40 epochs, batch size 128, weight decay 10^{-4}, standardized regression target; three hyperparameter configurations were tried during drafting, Table 4 reports the final one) are unchanged from an earlier draft; only the seeding protocol and the target-context input (Section 4) changed. Full hyperparameters, feature-column lists per strategy, and the exact chronological split boundary (2024-01-03 22:49) are recorded in the script itself and in `results/metrics.json`, both retained in the project repository alongside this manuscript so that every number in Sections 5–5.5 can be traced back to a specific, re-runnable computation rather than taken on the manuscript's word.

B. Feature lists by sequencing strategy

Strategy A (per-cell), 9 history features per timestep (i.e., of the target row's own K preceding observations for that cell):

- `active_users_roll3_cell`, `prb_utilization_pct`, `latency_ms`
- `handover_count`, `is_peak_hour`, `rsrp_rollvar5_cell`
- `time_since_last_s`, `slice_type_enc`, `cell_type_enc`

Strategy B (pooled), 8 history features per timestep (no elapsed-time feature, since Strategy B's pooled log has uniform one-minute global spacing by construction):

- `active_users_roll3`, `prb_utilization_pct`, `latency_ms`
- `handover_count`, `is_peak_hour`, `rsrp_rollvar5_global`
- `slice_type_enc`, `cell_type_enc`

Target context, 2 additional features per timestep, both strategies (the row actually being forecast's own identity, known at decision time, kept separate from the history window above – Section 4):

- `slice_type_enc` (target row), `cell_type_enc` (target row)

All three feature groups are standardized (zero mean, unit variance, statistics fit on the training split only) before being passed to any model; the history window is concatenated to a GBT model's flattened last-timestep vector or a GRU's sequence input, and the target context is concatenated separately, to GBT's flat vector or to the GRU's final hidden state, per Section 4.

References

[1] Mehdi Bennis, Mérouane Debbah, and H. Vincent Poor. Ultra-reliable and low-latency wireless communication: Tail, risk, and scale. *Proceedings of the IEEE*, 106(10):1834–1853, 2018.

[2] Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, 2014.

[3] Leandros Tassiulas and Anthony Ephremides. Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks. *IEEE Transactions on Automatic Control*, 37(12):1936–1948, 1992. doi: 10.1109/9.182479.

[4] Michael J. Neely. *Stochastic Network Optimization with Application to Communication and Queueing Systems*. Synthesis Lectures on Communication Networks. Morgan & Claypool Publishers, 2010.

[5] Leonidas Georgiadis, Michael J. Neely, and Leandros Tassiulas. Resource allocation and cross-layer control in wireless networks. *Foundations and Trends in Networking*, 1(1):1–144, 2006. doi: 10.1561/1300000001.

[6] Xenofon Foukas, Georgios Patounas, Ahmed Elmokashfi, and Mahesh K. Marina. Network slicing in 5G: Survey and challenges. *IEEE Communications Magazine*, 55(5):94–100, 2017. doi: 10.1109/MCOM.2017.1600951.

[7] Saloua Hendaoui, Fatma Hendaoui, and Nawel Zangar. Dynamic proactive–reactive scheduling for URLLC in 5G: Leveraging XGBoost and network virtualization. *Physical Communication*, 68:102553, 2025. doi: 10.1016/j.phycom.2024.102553.

[8] Niloofar Mehrnia, Gourav Prateek Sharma, Samie Mostafavi, Andreas Johnsson, Sinem Coleri, Carlo Fischione, and James Gross. AI-based KPI prediction methods in future 6G networks: A survey. *arXiv preprint arXiv:2606.01972*, 2026.

[9] Israel Tommy, Taoreed Akinola, Xiangfang Li, and Lijun Qian. Spatio-temporal beam-level traffic forecasting in 5G wireless systems using multi-task learning. *Frontiers in Communications and Networks*, 2025.

[10] Po-Heng Chou, Da-Chih Lin, Hung-Yu Wei, Walid Saad, and Yu Tsao. Measurement-driven early warning of reliability breakdown in 5G NSA railway networks. *arXiv preprint arXiv:2511.08851*, 2025.

[11] Wenhan Xu, Jiashuo Jiang, Lei Deng, and Danny Hin-Kwok Tsang. A Lyapunov drift-plus-penalty method tailored for reinforcement learning with queue stability. *arXiv preprint arXiv:2506.04291*, 2025.

[12] Ali Taleb Zadeh Kasgari and Walid Saad. Stochastic optimization and control framework for 5G network slicing with effective isolation. In *Proceedings of the 52nd Annual Conference on Information Sciences and Systems (CISS)*, 2018. arXiv:1801.10282.

[13] Narges Golmohammadi, Madan Mohan Rayguru, and Sabur Baidya. DRASTIC: A dynamic resource allocation framework over 6G network slicing in task-aware closed-loop tactile internet applications. *arXiv preprint arXiv:2603.27364*, 2026.

[14] Neco Villegas, Ana Larrañaga, Luis Díez, Katerina Koutlia, S. Lagén, and Ramón Agüero. Optimizing QoS MAC scheduling in 5G NR: A Lyapunov approach evaluated with XR traffic. *IEEE Transactions on Network and Service Management*, 2026. doi: 10.1109/TNSM.2025.3631520. Early access; volume and page numbers not yet assigned.

[15] Athanasios Tziouvaras, Carolina Fortuna, George Floros, Kostas Kolomvatsos, Panagiotis Sarigiannidis, Marko Grobelnik, and Blaž Bertalanič. Towards reliable AI in 6G: Detecting concept drift in wireless network. *arXiv preprint arXiv:2508.00042*, 2025.

[16] Omran Ayoub, Davide Andreoletti, Aleksandra Knapińska, Róża Goścień, Piotr Lechowicz, Tiziano Leidi, Silvia Giordano, Cristina Rottondi, and Krzysztof Walkowiak. Liquid neural network-based adaptive learning vs. incremental learning for link load prediction amid concept drift due to network failures. *arXiv preprint arXiv:2404.05304*, 2024.

[17] Ons Aouedi, Van An Le, Kandaraj Piamrat, and Yusheng Ji. Deep learning on network traffic prediction: Recent advances, analysis, and future directions. *ACM Computing Surveys*, 57(6):1–37, 2025. doi: 10.1145/3703447.

[18] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2017.

[19] João Gama, Indrė Žliobaitė, Albert Bifet, Mykola Pechenizkiy, and Abdelhamid Bouchachia. A survey on concept drift adaptation. *ACM Computing Surveys*, 46(4), 2014. doi: 10.1145/2523813.

[20] Haibo He and Edwardo A. Garcia. Learning from imbalanced data. *IEEE Transactions on Knowledge and Data Engineering*, 21(9):1263–1284, 2009. doi: 10.1109/TKDE.2008.239.

[21] Nitesh V. Chawla, Kevin W. Bowyer, Lawrence O. Hall, and W. Philip Kegelmeyer. SMOTE: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, 16:321–357, 2002. doi: 10.1613/jair.953.

[22] Srikumar Nayak. 5g network KPI dataset. Kaggle, n.d. srikumarnayak/5g-network-kpi-dataset.

[23] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[24] Jesse Davis and Mark Goadrich. The relationship between precision-recall and ROC curves. In *Proceedings of the 23rd International Conference on Machine Learning (ICML)*, pages 233–240, 2006.