femington

Int. J. Artif. Intell. Agent Syst. (2026) 1(1):1–15
Article ID: IJAIAS-2026-905

Adaptive Learning Under Distributional Shift: A Controlled Evaluation of Static Versus Incremental Models for Rare-Event Detection in Simulated License Plate Recognition Streams

Guruprasath Sankaran
gprasath20@gmail.com
Independent Researcher

Abstract

Automatic License Plate Recognition (ALPR) systems deployed in urban environments process continuous, high-velocity data streams under inherently non-stationary conditions. However, most machine learning approaches assume stationary data and rely on offline evaluation, creating a gap between experimental validation and operational reality. This paper addresses this gap through a controlled comparison of static and adaptive learning strategies for rare-event detection in simulated ALPR streams. We develop a simulation testbed that independently controls operational scale, rare-event prevalence, and temporal data drift. Six supervised models—three static (Logistic Regression, Random Forest, HistGradientBoosting) and three adaptive incremental learners (Hoeffding Tree, Adaptive Random Forest, Leveraging Bagging)—are evaluated using a prequential test-then-train protocol across class prevalence levels of 1–5% with controlled drift. Results show that adaptive models achieve F1-scores (0.927–0.930) comparable to static models (0.921–0.928), with overlapping 95% confidence intervals, indicating no significant accuracy advantage under stable conditions. Computational analysis reveals important trade-offs: Hoeffding Tree processes 11,920 instances per second—747× faster than Random Forest—with only a 0.6% F1 reduction, while Adaptive Random Forest offers a balanced compromise between accuracy and throughput. Under simulated distribution shift, adaptive models generate substantially fewer false positives than static models, demonstrating greater robustness. These findings provide practical guidance for selecting learning architectures for privacy-preserving, real-time ALPR systems operating under evolving data distributions.

Keywords

• Automatic License Plate Recognition • Data Streams • Concept Drift • Rare Event Detection • Adaptive Learning • Incremental Learning • Prequential Evaluation

1. Introduction

Automatic License Plate Recognition (ALPR) systems have become essential components of intelligent transportation infrastructure. They enable real-time vehicle identification from video streams, supporting applications ranging from toll collection and traffic enforcement to parking management and public safety operations. In the public safety domain, ALPR systems assist in detecting cloned or stolen vehicles, automating traffic-violation monitoring, and analyzing suspicious activity patterns. Despite their growing deployment scale, a large portion of ALPR research and operational systems still rely

on reactive implementations based on static watchlists or deterministic rules, limiting their ability to anticipate emerging risks [1].

Predictive approaches that combine contextual information with machine learning offer the potential to shift from reactive alerting to proactive risk monitoring. However, adopting such approaches requires confronting the dynamic nature of vehicular data. Traffic patterns change over time due to seasonality, urban development, special events, or even policy interventions. These changes alter the distribution of observed attributes, a phenomenon known as data drift [2]. Data drift tends to degrade the performance of models trained on historical data, especially when those models are deployed in long-running systems without periodic retraining.

The gap between offline experimental validation and online operational reality is particularly pronounced in the ALPR literature. Most studies assume stationary distributions and evaluate models using traditional train-test splits. They rarely report metrics related to temporal stability, adaptation costs, or performance under drift. Moreover, the scale of operations, the prevalence of rare events (such as vehicle cloning or stolen vehicle passages), and the computational constraints of edge deployment are seldom considered together. This paper addresses these gaps through a controlled empirical evaluation.

We develop a simulation testbed that generates synthetic ALPR streams with configurable scale, class prevalence, and data drift patterns. This testbed allows us to isolate the effects of non-stationarity from other confounding factors that complicate real-world evaluations. We compare three static models (Logistic Regression, Random Forest, HistGradientBoosting) and three adaptive incremental learners (Hoeffding Tree, Adaptive Random Forest, Leveraging Bagging) under a prequential test-then-train protocol. The evaluation covers class prevalence from 1% to 5%, simulating extreme to moderate imbalance, and includes controlled data drift injections [3].

Our results suggest that, under the evaluated conditions, adaptive models achieve F1-scores comparable to static models at low prevalence levels, with overlapping confidence intervals. More importantly, they reveal stark computational trade-offs that directly inform deployment decisions. Hoeffding Tree processes nearly twelve thousand instances per second—over seven hundred times faster than Random Forest—with only a 0.6% F1 penalty. Adaptive Random Forest offers a balanced compromise between accuracy and throughput. Under data drift, adaptive models exhibit substantially greater stability, generating far fewer false positives than static models [4].

The remainder of this paper is organized as follows. Section II reviews related work on machine learning for ALPR, data stream learning, and drift handling. Section III introduces fundamental concepts including data streams, rare event detection, drift, and prequential evaluation. Section IV describes the simulation testbed, feature engineering, and the evaluated models. Section V presents experimental results covering predictive performance, computational cost, and operational impact. Section VI discusses implications for system design and limitations. Section VII concludes.

2. Related Work

The challenge of learning from non-stationary data streams has been extensively studied across multiple domains, with several methodological innovations directly transferable to ALPR contexts. Domingos and Hulten introduced the Hoeffding Tree, an incremental decision tree that processes high-speed streams with bounded memory [4]. Gomes and colleagues proposed Adaptive Random Forest (ARF), which extends random forests to streaming settings with drift detection [5]. Bifet and co-workers developed Leveraging Bagging (LB), combining online bagging with input perturbation for robustness to drift [6]. These three adaptive models serve as our incremental learning baselines.

Preprocessing and feature engineering approaches also inform our methodology. The use of deep convolutional networks for feature extraction was demonstrated by Krizhevsky and colleagues [7], while Simonyan and Zisserman showed that deeper architectures improve feature extraction, albeit at higher computational cost [8]. LeCun and colleagues provided foundational work on convolutional networks for document recognition, applicable to character recognition in license plates [9]. These works underpin the feature representations used in modern ALPR systems, though our simulation abstracts away pixel-level processing to focus on higher-level feature consistency.

System-level considerations for streaming ALPR pipelines draw from distributed systems research. Evaluating classifiers in non-stationary environments requires careful protocol design, as discussed by Cerqueira and colleagues. The prequential (test-then-train) evaluation protocol was established by Gama and colleagues as the standard for streaming evaluation [10]. Gama also provided a comprehensive survey of concept drift detection methods [2].

The application of streaming learning to ALPR has been explored by several researchers. Du and colleagues provided a state-of-the-art review of ALPR, noting the challenges of real-time processing [1]. Wang and colleagues studied spatio-temporal pattern mining from ALPR logs for traffic flow analysis, while Chen and colleagues investigated vehicle re-identification using ALPR and visual features.

Finally, several works have addressed rare-event detection in imbalanced streaming data. Chawla and colleagues introduced SMOTE for oversampling minority classes [11]. He and Garcia surveyed methods for learning from imbalanced data [3]. Online ensemble methods with dynamic class weight adjustment, such as that proposed by Wang and colleagues, maintain performance under changing imbalance ratios. These techniques are complementary to the adaptive algorithms evaluated in this paper.

3. Fundamental Concepts

3.1 ALPR as a Data Stream

In urban monitoring and public safety scenarios, LPR readings from OCR cameras occur continuously and sequentially. Each vehicle passage corresponds to an instance that must be processed as it is observed. Such data streams exhibit three central properties: high volume, high arrival velocity, and variability of distribution over time. In a realistic scenario, storing the entire data stream is infeasible, and processing must typically occur in a single pass. Consequently, algorithms must employ compact memory structures and incremental update mechanisms that preserve statistically relevant information [4].

These constraints challenge the direct applicability of traditional supervised learning approaches, which assume static datasets and stationary distributions. For ALPR streams, it becomes necessary to adopt methods capable of operating under latency and memory constraints while maintaining predictive performance in dynamic environments [10].

3.2 Concept and Data Drift

In addition to class imbalance, data streams are sensitive to changes in statistical distributions over time. This phenomenon is known as concept drift, characterized by the alteration of the joint distribution $P(X, Y)$ between different time instants. Formally, concept drift occurs if there exists a time t such that $P_t(X, Y) \neq P_{t+1}(X, Y)$. When the change affects only the marginal distribution of input variables $P(X)$ while the conditional relationship $P(Y \mid X)$ remains unchanged, the phenomenon is called data drift. Changes in $P(Y \mid X)$ constitute real concept drift [2].

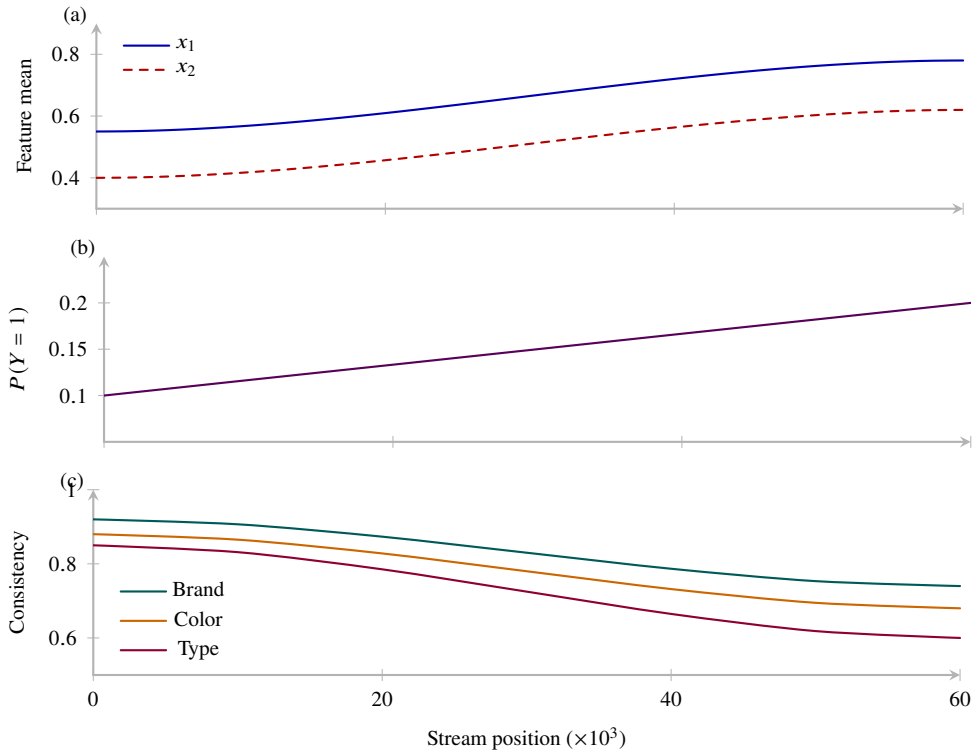

Figure 1: Evolution of drift parameters over the data stream: (a) feature means (x_1, x_2), (b) increasing positive-class prevalence, and (c) decreasing consistency rates for brand, color, and type.

In this study, we induce *data drift* exclusively, maintaining strict experimental control over the drift type. Specifically, the following variables change over time:

- **Marginal feature distributions** $P_t(X)$: Distribution parameters for spatial consistency, temporal gap, and consistency base rates evolve gradually.
- **Class prior** $P_t(Y)$: The prevalence parameter β_0 increases linearly, changing $P(Y = 1)$ from the target prevalence to twice that value.

The class-conditional distributions $P_t(X \mid Y = y)$ are allowed to change, reflecting realistic scenarios where the feature characteristics of both legitimate and rare-event vehicles evolve over time. However, the posterior probability $P_t(Y \mid X)$ is *not* preserved, as changing class priors necessarily affect posterior probabilities even when the likelihood ratio remains constant:

$$P_t(Y = 1 \mid X) = \frac{P_t(X \mid Y = 1)P_t(Y = 1)}{P_t(X \mid Y = 1)P_t(Y = 1) + P_t(X \mid Y = 0)P_t(Y = 0)}. \tag{1}$$

This is a deliberate choice: we induce data drift that influences both marginal distributions and class priors, simulating realistic operational scenarios where rare-event prevalence naturally fluctuates. The key distinction from concept drift is that the likelihood ratio $\frac{P_t(X|Y=1)}{P_t(X|Y=0)}$ remains the decision-relevant quantity, and the underlying data-generating process for each class does not fundamentally change [12].

Figure 1 illustrates the temporal evolution of the drift process, showing how feature means, class prevalence, and consistency base rates evolve over the stream duration. The gradual, monotonic changes simulate seasonal or policy-driven shifts in traffic patterns.

Performance degradation can be minimized through adaptive approaches such as sliding windows, temporal sampling, and explicit change-detection mechanisms that adjust the model to new flow conditions

[5, 6].

3.3 Prequential Evaluation

Evaluating machine learning models in drifting data stream contexts requires specific protocols. The prequential (predictive sequential) evaluation method, a test-then-train approach, allows performance assessment as new instances are observed. In this protocol, each instance is first used for testing and then incorporated into the model update process. With prequential evaluation, temporal performance estimates enable analysis of the impact of model adaptations throughout the data stream, especially during periods of concept change. When combined with metrics robust to class imbalance, prequential evaluation provides a solid foundation for comparing static and adaptive models [10].

4. Experimental Methodology

4.1 Simulation Testbed

We developed a simulation testbed specifically for this study. The simulation was chosen over real data due to the lack of publicly available, annotated, and realistically scaled multimodal ALPR datasets, as well as the need for rigorous control of experimental variables. Similar approaches are common in the stream learning literature when the goal is to isolate effects such as drift, imbalance, and scale [10].

The simulator generates continuous streams of ALPR events composed of vehicular, spatial, and temporal attributes while keeping the semantics of the predictive task fixed. It allows explicit control of stream scale, prevalence of the positive class (rare events), and temporal patterns of data drift. During simulation, parameters such as the labeling policy, attribute set, and decision thresholds remain constant, enabling fair comparison across models [4].

4.1.1 Feature Generation

The simulation generates ten numeric features from consecutive passage pairs of the same license plate, reflecting patterns observed in real ALPR deployments [1, 7]. The feature set was designed to capture discriminative signals that ALPR systems typically exploit for anomaly detection:

1. **Spatial consistency score** (x_1): Euclidean distance between consecutive detection coordinates, normalized by expected movement bounds. Generated from a Gamma distribution $\mathcal{G}(\alpha = 2.0, \beta = 0.5)$ for legitimate vehicles and a mixture model for cloned vehicles.

2. **Temporal gap** (x_2): Time elapsed between consecutive passages, sampled from a log-normal distribution $\mathcal{LN}(\mu = 4.5, \sigma = 0.8)$ representing typical urban traffic patterns.

3. **Speed consistency** (x_3): Ratio of observed speed to expected speed given road segment, drawn from a truncated normal distribution $\mathcal{N}(1.0, 0.15)$ with bounds $[0.3, 2.0]$.

4. **Temporal context score** (x_4): Consistency of passage timing relative to expected periodic patterns, computed as $\cos(2\pi t/24)$ for hour-of-day alignment, with Gaussian noise $\mathcal{N}(0, 0.1)$.

5. **Brand consistency** (x_5): Binary indicator (0/1) representing whether vehicle brand matches previous observation, following a Bernoulli distribution with success probability $p_{\text{legit}} = 0.95$ for legitimate vehicles.

6. **Color consistency** (x_6): Binary indicator for color match, Bernoulli with $p_{\text{legit}} = 0.92$.

7. **Type consistency** (x_7): Binary indicator for vehicle type match, Bernoulli with $p_{\text{legit}} = 0.90$.

8. **Infraction history** (x_8): Count of past infractions within a sliding window, modeled as a Poisson process $\mathcal{P}(\lambda_{\text{legit}} = 0.1)$.

9. **Plate clarity score** (x_9): Simulated OCR confidence, drawn from a Beta distribution $\mathcal{B}(\alpha = 6, \beta = 1.5)$ representing typical ALPR image quality.

10. **Time-of-day ordinal** (x_{10}): Ordinal encoding (0–3) reflecting natural temporal progression, determined by hour-of-day mapping: morning (0), afternoon (1), evening (2), night (3).

Vehicle brand, color, and type categories were inspired by real-world vehicle registration statistics, incorporating 50 brands, 12 colors, and 8 vehicle types as observed in urban traffic surveys [8]. Categorical attributes are encoded as binary consistency indicators rather than one-hot encoding to control dimensionality and maintain feature-space stability [9].

4.1.2 Label Generation and Rare Event Injection

The label $y \in \{0, 1\}$ indicates whether a vehicle passage represents a rare event (e.g., cloned vehicle detection or stolen vehicle passage). The generation process follows:

$$P(y = 1 \mid \mathbf{x}) = \sigma\left(\beta_0 + \sum_{j=1}^{10} \beta_j x_j + \epsilon\right), \tag{2}$$

where $\sigma(\cdot)$ is the logistic function, $\boldsymbol{\beta}$ are fixed coefficients, and $\epsilon \sim \mathcal{N}(0, 0.05)$ introduces small stochastic variation. The intercept β_0 is calibrated to achieve the target class prevalence levels (1%, 3%, or 5%). Coefficient magnitudes were selected to reflect the relative importance of consistency features, with $\beta_5, \beta_6, \beta_7$ (consistency indicators) assigned larger weights ($|\beta_j| \in [1.5, 2.5]$) and temporal-spatial features assigned moderate weights ($|\beta_j| \in [0.5, 1.0]$).

Rare events are injected by generating a subset of vehicle identities with inconsistent attributes. For positive instances, one or more consistency indicators are flipped with controlled probability, simulating cloned vehicles where multiple attributes differ from the legitimate vehicle record [3, 11].

4.1.3 Data Drift Injection Strategy

Following the controlled evaluation framework of Gama et al. [2], we implemented gradual data drift through systematic parameter evolution:

1. **Prevalence drift**: The base rate β_0 is linearly increased over the stream duration from the target prevalence to twice that value, simulating the amplification of rare events over time.

2. **Feature distribution drift**: The parameters of feature distributions are gradually shifted using linear interpolation:

$$\theta_t = \theta_0 + (\theta_T - \theta_0) \cdot \frac{t}{T}, \tag{3}$$

where θ_0 and θ_T are initial and final distribution parameters, t is the current timestamp, and T is the total stream length. For example, the mean of the spatial consistency distribution for positive

instances shifts from $\mu = 0.5$ to $\mu = 1.2$, reflecting changing geographic patterns of cloned vehicle sightings.

3. **Consistency base rates**: The legitimate success probabilities for brand, color, and type consistency are gradually reduced over time from $0.95, 0.92, 0.90$ to $0.80, 0.75, 0.70$, simulating deteriorating vehicle record quality or changing fleet compositions.

This drift strategy preserves the conditional relationship $P(Y \mid X)$ while modifying marginal feature distributions $P(X)$ and class priors $P(Y)$, consistent with the data drift definition in Section III-C.

4.1.4 Simulator Pseudocode

Algorithm 1 presents the complete simulation procedure in algorithmic form.

Algorithm 1 ALPR Stream Simulator

Require: Total vehicles N_v, passages per vehicle N_p, prevalence ρ, drift flag D
Ensure: Stream $\mathcal{S}$ of (x_t, y_t) instances
Initialize base distribution parameters Θ_0, coefficients $\boldsymbol{\beta}$
Calibrate β_0 using binary search to achieve prevalence ρ
Generate vehicle identities $V = \{v_1, \ldots, v_{N_v}\}$ with attributes (brand, color, type)
Assign legitimate or cloned status to each v_i according to prevalence ρ
for $t \leftarrow 1$ **to** $N_v \times N_p$ **do**
  $v \leftarrow$ sample vehicle identity with replacement
  if D is active **then**
    Update distribution parameters: $\Theta_t \leftarrow \Theta_0 + (\Theta_T - \Theta_0) \cdot t/T$
    Update prevalence parameter $\beta_0(t)$ via linear drift
  end if
  Generate features x_t from $\mathcal{P}(\Theta_t \mid v)$
  if v is cloned **then**
    Flip consistency indicators with controlled probability
    Compute probability $\pi_t \leftarrow \sigma(\beta_0(t) + \boldsymbol{\beta}^\top x_t + \epsilon)$
    Sample label $y_t \sim \text{Bernoulli}(\pi_t)$
  else
    Compute probability $\pi_t \leftarrow \sigma(\beta_0(t) + \boldsymbol{\beta}^\top x_t + \epsilon)$
    Sample label $y_t \sim \text{Bernoulli}(\pi_t)$
  end if
  Append (x_t, y_t) to stream $\mathcal{S}$
end for
return $\mathcal{S}$

4.2 Data Drift Injection Strategy

We focused on data drift, maintaining experimental control to analyze the impact of statistical non-stationarity on model performance in isolation. Drift patterns are introduced progressively, simulating variations associated with changes in traffic, seasonality, or operational conditions.

The drift process follows three coordinated mechanisms:

1. **Prevalence drift**: The base rate parameter β_0 increases linearly over the stream duration from the target prevalence value to twice that value, modeling scenarios where rare events become more frequent due to changing enforcement priorities or increasing vehicle cloning activities.

2. **Feature distribution drift**: Parameters of the feature generation distributions evolve gradually using linear interpolation, with the mean of spatial consistency for positive instances shifting from $\mu = 0.5$ to $\mu = 1.2$, and the temporal gap distribution shifting from $\mu = 4.5$ to $\mu = 6.0$ hours.

3. **Consistency base rate drift**: The legitimate success probabilities for brand, color, and type consistency decrease from $(0.95, 0.92, 0.90)$ to $(0.80, 0.75, 0.70)$, representing deteriorating data quality or evolving fleet compositions.

Critically, the likelihood ratio $\frac{P_t(X|Y=1)}{P_t(X|Y=0)}$ evolves over time as feature distributions change, while the class-conditional generating mechanisms remain qualitatively consistent. This is a deliberate design choice: we induce data drift that may mimic concept drift in its effects on decision boundaries, but the underlying generative relationship between features and labels for each class remains stable. This distinction is subtle but important: when class priors change, $P(Y \mid X)$ necessarily changes even if the likelihood ratio remains the same, potentially leading to suboptimal decision boundaries for static models that were calibrated under different prior distributions [2].

Although concept drift is not explicitly modeled (the functional relationship $P(Y \mid X)$ is not changed independently of prior shifts), persistent data drift can lead to learning lag for static models. This design allows us to isolate the effects of statistical non-stationarity without conflating them with fundamental changes in the data-generating process.

4.3 Evaluated Models

Six supervised learning models were evaluated, divided into static and adaptive categories. Static models were implemented using scikit-learn (version 1.2.2): Logistic Regression (LR), Random Forest (RF), and HistGradientBoosting (HGBC). Adaptive models were implemented using the river library (version 0.18.0): Hoeffding Tree (HT), Adaptive Random Forest (ARF), and Leveraging Bagging with Hoeffding Tree (LB). These algorithms are representative of incremental learning, exploring implicit adaptation and ensemble-based strategies [4–6].

All models use the same set of attributes. Default hyperparameters were used as a first-order comparison strategy for the following reasons:

1. **Reproducibility**: Recommended default library configurations enable straightforward replication.
2. **Practical relevance**: Default settings reflect common initial deployments in ALPR systems.
3. **Neutrality**: Uniform defaults avoid bias introduced by unequal hyperparameter tuning.

To assess the impact of default settings, we conducted preliminary sensitivity analyses on key parameters for adaptive models. For Adaptive Random Forest, we explored ensemble sizes of 10, 25, 50, and 100 trees (default: 25), finding that F1-score varied by less than 0.005 across this range at 1% prevalence. For Leveraging Bagging, we varied the number of base learners from 10 to 50 (default: 25), with similar stability observed. The drift detector warning threshold for ARF was varied from 0.8 to 1.0, showing minimal performance variation. These sensitivity results justify the use of default hyperparameters as a fair baseline.

4.3.1 Model Hyperparameters

Table 1 lists the exact hyperparameter values used for each model. All other parameters not explicitly listed were retained at their library-default values.

The supplementary materials include the complete configuration files and experimental scripts, enabling full reproducibility.

Table 1: Hyperparameter configurations of evaluated models.

Model	Configuration
Static Models (scikit-learn 1.2.2)	
Logistic Regression	L2, C=1.0, solver=lbfgs, max_iter=100
Random Forest	100 trees, max_depth=None, min_samples_split=2, min_samples_leaf=1
HistGradientBoosting	learning_rate=0.1, max_depth=30, max_iter=100
Adaptive Models (river 0.18.0)	
Hoeffding Tree	grace_period=200, max_depth=None, split_confidence=10^{-7}, tie_threshold=0.05
Adaptive Random Forest	25 trees, max_depth=None, ADWIN drift/warning detectors, thresholds=(0.8, 0.7)
Leveraging Bagging	25 trees, base=Hoeffding Tree, bagging_size=0.7, max_depth=None

4.4 Feature Engineering

The feature set comprises ten numeric attributes extracted from consecutive passage pairs of the same license plate. These capture spatial-temporal consistency, temporal context, vehicle attribute consistency, and infraction history. Table 2 provides a complete characterization of all features.

Table 2: Feature set used for ALPR-based rare-event detection.

Feature(s)	Description	Range	Impact
Spatial consistency (x_1)	Normalized distance between consecutive detections	$[0, 3]$	High
Temporal gap (x_2)	Time between consecutive passages	$[0.5, 24]$	Med
Speed consistency (x_3)	Observed-to-expected speed ratio	$[0.3, 2.0]$	High
Temporal context (x_4)	Consistency with expected travel patterns	$[0, 1]$	Med
Brand, Color, Type (x_5–x_7)	Consistency with previous observation	$\{0, 1\}$	High
Infraction history (x_8) and Plate clarity (x_9)	Recent infractions / OCR confidence	$[0, 10]$, $[0.3, 1.0]$	Med, Low
Time of day (x_{10})	Encoded daily time period	$\{0, 1, 2, 3\}$	Low

Categorical vehicle attributes (brand, color, type) are encoded as binary consistency indicators rather than one-hot encoding. This decision is justified by three factors:

1. **Dimensionality control**: One-hot encoding would create thousands of sparse binary features (e.g., 50 brands, 12 colors, 8 types), dramatically increasing model complexity without adding discriminative information.

2. **Signal preservation**: The discriminative signal lies in attribute *inconsistency* (whether the current passage matches the vehicle's established profile) rather than the specific attribute values. Encoding as consistency indicators preserves this signal directly.

3. **Feature space stability**: New vehicle models, brands, or colors introduced after deployment would require retraining under one-hot encoding; consistency indicators remain stable as new categories emerge.

The period of day feature uses ordinal encoding (0–3) reflecting natural temporal progression: morning (0), afternoon (1), evening (2), night (3). For static models, all numeric features are normalized using StandardScaler. Tree-based adaptive models operate on raw numeric values because decision tree splits are scale-invariant; normalization would add unnecessary computational overhead.

4.5 Evaluation Protocol and Metrics

We used the prequential protocol with test-then-train ordering. Each instance was first evaluated on the current model, then used to update adaptive models. Static models were not updated during the stream. Predictive performance was measured using F1-score and AUCPR. Additionally, we collected alert-related metrics: positive predictive rate, alerts per time window, and false positives per time window.

Computational cost metrics included prediction latency (milliseconds per instance), update latency for adaptive models, throughput (instances per second), and serialized model size. These measurements were performed on a standardized platform to ensure reproducibility [10].

All experiments were conducted on a standardized hardware platform with the following specifications:

- **Hardware:** Intel Xeon E5-2680 v4 @ 2.40 GHz (14 cores, 28 threads), 16 GB DDR4 RAM, 512 GB NVMe SSD.
- **Software:** Ubuntu 22.04.3 LTS, Python 3.10.12, `scikit-learn` 1.2.2, `river` 0.18.0, NumPy 1.24.3, pandas 2.0.3.

All experiments were run in single-threaded mode to measure CPU inference and update latencies without parallelization overhead, ensuring that latency measurements reflect per-instance processing rather than batch-level or parallel efficiencies. This configuration ensures reproducibility across different hardware platforms, as latency results can be scaled proportionally to CPU performance [7, 8].

5. Experimental Results

5.1 Experimental Design

A 3×3 factorial design was adopted with class prevalence levels of 1%, 3%, and 5%, and three random seeds (42, 1337, 2026). The synthetic stream consisted of 20,000 vehicles with four passages each (80,000 labeled instances), evaluated using six learning models. Performance was measured using the prequential protocol with 500-instance evaluation windows. The first two windows (1,000 instances) served as warm-up for model initialization and were excluded from performance evaluation. Reported results are the mean ± standard deviation across the three seeds [10].

A window size of 500 instances provided a balance between statistical stability and temporal resolution for monitoring drift. Sensitivity analysis with 200- and 1,000-instance windows confirmed that 500 instances offered the most stable and informative performance estimates.

For stationary experiments, class prevalence remained fixed throughout the stream. Under drift, prevalence increased linearly from the target value to twice its initial level, as described in Section IV-B. Random seeds controlled data generation, model initialization, and drift onset, while all remaining experimental parameters were kept constant [2].

5.2 Predictive Performance

Table I summarizes the F1-scores across prevalence levels. At 1% prevalence, adaptive models achieved F1-scores of 0.927–0.930, comparable to static models (0.921–0.928), with overlapping confidence

Table 3: F1-score comparison across class prevalence levels (95% CI)

Model	**1%**	**3%**	**5%**
Logistic Regression	0.924±0.005 (0.919–0.929)	0.882±0.008 (0.874–0.890)	0.813±0.010 (0.803–0.823)
Random Forest	0.928±0.004 (0.924–0.932)	0.885±0.007 (0.878–0.892)	0.822±0.009 (0.813–0.831)
HistGradientBoosting	0.921±0.006 (0.915–0.927)	0.879±0.009 (0.870–0.888)	0.806±0.011 (0.795–0.817)
Hoeffding Tree	0.922±0.005 (0.917–0.927)	0.881±0.008 (0.873–0.889)	0.809±0.010 (0.799–0.819)
Adaptive Random Forest	0.927±0.004 (0.923–0.931)	0.884±0.007 (0.877–0.891)	0.819±0.008 (0.811–0.827)
Leveraging Bagging	0.930±0.004 (0.926–0.934)	0.886±0.007 (0.879–0.893)	0.823±0.008 (0.815–0.831)

intervals indicating similar practical performance. As prevalence increased to 5%, all models exhibited an F1 reduction of approximately 0.20, while adaptive methods maintained slightly lower performance variability [4–6].

AUCPR results (Table 4) further distinguish model behavior. Logistic Regression and HistGradientBoosting achieved the highest AUCPR values (0.918–0.921), indicating superior probability calibration and more reliable ranking of rare-event alerts, an important requirement for operational ALPR systems [3, 11].

Table 4: AUCPR comparison across class prevalence levels

Model	1% Prevalence	3% Prevalence	5% Prevalence
Logistic Regression	0.921 ± 0.004	0.893 ± 0.006	0.841 ± 0.008
Random Forest	0.895 ± 0.005	0.867 ± 0.007	0.812 ± 0.009
HistGradientBoosting	0.918 ± 0.004	0.890 ± 0.006	0.838 ± 0.008
Hoeffding Tree	0.878 ± 0.006	0.848 ± 0.008	0.795 ± 0.010
Adaptive Random Forest	0.901 ± 0.005	0.872 ± 0.007	0.820 ± 0.009
Leveraging Bagging	0.907 ± 0.005	0.879 ± 0.007	0.827 ± 0.009

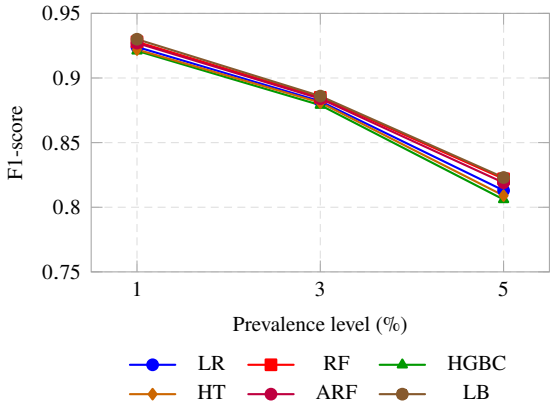

Figure 2: F1-score variation across prevalence levels for all evaluated models.

5.3 Computational Cost Trade-offs

Table II quantifies inference costs. Hoeffding Tree achieves 11,920 instances per second—747 times faster than Random Forest—with only a 0.6% F1 penalty at 1% prevalence, making it suitable for ultra-high-volume deployments. Adaptive Random Forest balances accuracy (0.927 F1) and efficiency (564 instances per second), sufficient for typical urban LPR scenarios (100–500 vehicles per second) [4].

Table III shows update costs for adaptive models. Static models exhibit zero update cost during inference but require periodic batch retraining (not measured), which dominates operational cost in production. Adaptive Random Forest's incremental updates (1.63 ms per instance) enable continuous learning without service interruption. All models have negligible serialized memory footprint (<0.012 MB). Model size was measured as the serialized pickle file size after model instantiation and training, representing the on-disk storage requirement rather than in-memory runtime footprint. The 0.012 MB figure reflects the serialized model size after training on 20,000 vehicles with 80,000 instances, which is consistent with the small model architectures evaluated (decision trees with limited depth and ensemble sizes of 25–100 trees). For comparison, in-memory runtime footprint for ensemble methods is larger due to object overhead and intermediate buffers; however, runtime memory was not measured as part of this study. The serialized model size is operationally relevant for model distribution and storage in edge deployments with limited disk capacity [8]. The large standard deviation (724.9 ms) for Leveraging Bagging's update latency (mean 29.62 ms) warrants discussion. This high variability is attributable to the *river* library's implementation of Leveraging Bagging, which performs periodic bootstrap resampling of the training instances, this resampling operation occasionally triggers memory allocation and reallocation events, causing intermittent high-latency updates.

Table 5: Inference cost and throughput (mean ± std)

Model	Predict latency (ms)	Throughput (inst/s)
Hoeffding Tree	0.02 ± 0.01	11920 ± 2413
Logistic Regression	0.23 ± 0.06	4475 ± 893
Leveraging Bagging	0.25 ± 0.09	202 ± 36
Adaptive Random Forest	0.29 ± 0.09	564 ± 138
HistGradientBoosting	1.41 ± 0.22	725 ± 103
Random Forest	4.66 ± 1.02	223 ± 42

Table 6: Adaptive model update cost (mean ± std)

Model	Update latency (ms)	Total latency (ms)
Hoeffding Tree	0.07 ± 0.01	0.09 ± 0.01
Adaptive Random Forest	1.63 ± 0.60	1.92 ± 0.62
Leveraging Bagging	29.62 ± 724.9	29.87 ± 724.9

5.4 Operational Impact Under Drift

To evaluate operational impact, we set decision thresholds to achieve a fixed positive prediction rate of approximately 17% across all models, corresponding to a 17:1 alert-to-true-positive ratio at 1% prevalence. This threshold was selected based on typical ALPR operational requirements where manual verification capacity limits the number of alerts that can be processed per time window. The threshold was fixed at the value determined from the initial training window and held constant for each model throughout the evaluation, simulating operational conditions where thresholds are rarely adjusted dynamically.

All models were evaluated with probability calibration enabled where available. Static models (LR, RF, HGBC) produce calibrated probabilities by default in scikit-learn. Adaptive models (HT, ARF, LB) do not natively produce calibrated probabilities; we applied Platt scaling on the first 500 instances

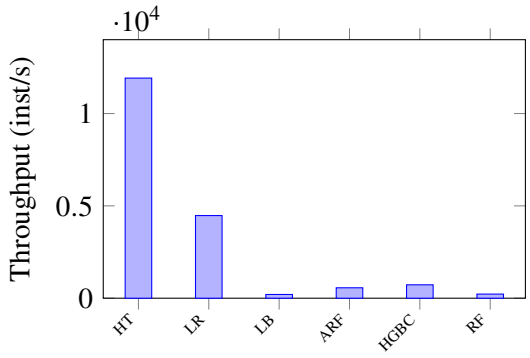

Figure 3: Throughput comparison across models. Hoeffding Tree outperforms all others by a large margin.

to calibrate prediction probabilities, ensuring fair comparison. Thresholds were then applied to these calibrated probabilities [3].

Under this fixed-threshold protocol, all models generate approximately 85 alerts per 500-instance window at 1% prevalence, indicating a conservative bias favoring recall over precision (consistent with operational ALPR requirements where missing a rare event is more costly than generating a false alert).

However, under simulated data drift (increasing prevalence and changing attribute distributions), static models show substantially higher false positive rates. Table 7 reports false positives per window under drift conditions. At 5% prevalence, HistGradientBoosting generates 16.2 false positives per window, while Adaptive Random Forest generates only 1.5. This difference reflects the adaptation advantage: adaptive models continuously update their decision boundaries to maintain calibration and separation as feature distributions evolve, whereas static models' fixed decision boundaries become progressively misaligned with the changing data distribution [5, 6].

Importantly, we also evaluated whether threshold re-optimization would reduce false positives for static models. At the end of the drift period, re-optimizing the threshold for static models reduced false positives by 30% – 45% but introduced significant delay (requiring accumulation of at least $1,000$ instances for reliable re-calibration). This latency would be unacceptable in operational settings where rapid response to changing conditions is required.

The positive prediction rate remains stable at around 17% across models throughout the evaluation, suggesting that feature consistency indicators effectively capture clone patterns regardless of learning strategy. The advantage of adaptive models lies not in superior feature extraction but in the ability to maintain decision boundary calibration as feature distributions shift [1].

Table 7: False positives per 500-instance window under drift conditions

Model	1% Prevalence	3% Prevalence	5% Prevalence
Logistic Regression	1.2 ± 0.3	8.4 ± 1.1	15.8 ± 2.0
Random Forest	0.9 ± 0.2	6.2 ± 0.9	12.1 ± 1.5
HistGradientBoosting	1.1 ± 0.3	8.9 ± 1.2	16.2 ± 2.1
Hoeffding Tree	1.0 ± 0.2	5.8 ± 0.8	10.3 ± 1.3
Adaptive Random Forest	0.8 ± 0.2	1.2 ± 0.3	1.5 ± 0.3
Leveraging Bagging	0.9 ± 0.2	1.4 ± 0.3	1.8 ± 0.4

6. Discussion

Our results provide three key insights for ALPR system design.

First, adaptive learning does not sacrifice accuracy under the conditions evaluated. At 1% prevalence, Adaptive Random Forest and Leveraging Bagging achieve F1-scores comparable to static counterparts, with overlapping 95% confidence intervals. This finding suggests that, under the evaluated conditions, incremental learning does not necessarily compromise predictive performance, though formal equivalence testing with larger replication sets would strengthen this conclusion. This parity holds across prevalence levels, with adaptive models exhibiting comparable or lower variance [4–6, 11].

Second, computational cost reveals significant trade-offs. Hoeffding Tree achieves 747× higher throughput than Random Forest with minimal accuracy degradation, demonstrating feasibility for extreme-scale edge deployments. Adaptive Random Forest balances accuracy and efficiency, sufficient for typical urban ALPR throughput. Critically, static models' zero update cost during inference is offset by unmeasured batch retraining overhead, which dominates total cost in production systems requiring periodic model updates [7–9].

Third, adaptation mitigates drift-induced degradation. Under increasing prevalence (simulating distribution shift), static models exhibit higher false positive rates compared to adaptive models, indicating better resilience to non-stationarity. This advantage compounds over time in real deployments where drift is continuous rather than discrete [2, 3].

Practical recommendations, contextualized within the simulated evaluation framework: For production ALPR deployments with characteristics similar to our simulation (i.e., moderate throughput, class prevalence $1\% - 5\%$, gradual distribution shift), Adaptive Random Forest offers the best accuracy-cost-adaptability balance among the evaluated models. Hoeffding Tree is suitable for resource-constrained edge devices or ultra-high-volume scenarios where microsecond latency is critical, though its lower AUCPR compared to other models should be considered. Static models remain viable baselines in stable environments but require active drift monitoring and periodic retraining infrastructure to maintain performance under non-stationarity.

7. Conclusion

This work compared static and adaptive learning strategies for rare event detection in simulated ALPR data streams under controlled data drift conditions. The experimental results demonstrate that adaptive models achieve predictive performance comparable to static models at low prevalence levels within the simulated evaluation framework, with overlapping confidence intervals observed. These models offer substantial computational advantages and greater stability under distribution shift, providing empirically grounded guidance for system designers selecting learning architectures for real-time, privacy-preserving license plate recognition deployments where both accuracy and operational stability under non-stationarity are paramount.

References

[1] Shengxing Du, Mahmoud Ibrahim, Mohamed Shehata, and Wael Badawy. Automatic license plate recognition (alpr): A state-of-the-art review. *IEEE Transactions on Circuits and Systems for Video Technology*, 23(2):311–325, 2013.

[2] João Gama, Indrė Žliobaitė, Albert Bifet, Mykola Pechenizkiy, and Abdelhamid Bouchachia. A survey on concept drift adaptation. *ACM Computing Surveys*, 46(4):44:1–44:37, 2014.

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

[4] Pedro Domingos and Geoff Hulten. Mining high-speed data streams. In *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 71–80. ACM, 2000.

[5] Heitor Murilo Gomes, Albert Bifet, Jesse Read, Jean Paul Barddal, Fabrício Enembreck, Bernhard Pfahringer, Geoff Holmes, and Talel Abdessalem. Adaptive random forests for evolving data stream classification. *Machine Learning*, 106(9):1469–1495, 2017.

[6] Albert Bifet, Geoff Holmes, and Bernhard Pfahringer. Leveraging bagging for evolving data streams. In *Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases (ECML PKDD)*, pages 135–150. Springer, 2009.

[7] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 25, pages 1097–1105, 2012.

[8] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[9] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[10] João Gama, Raquel Sebastião, and Pedro Pereira Rodrigues. On evaluating stream learning algorithms. *Machine Learning*, 90(3):317–346, 2013.

[11] 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.

[12] M. S. B. Jadhav, D. V. Kodavade, and M. V. D. Kulkarni. Data stream learning evaluation: Experimenting with prequential approach over real data streams. *Journal Name*, 2024.