Adaptive Dynamic Feature Selection using Evolutionary Meta-Learning for Streaming Data

(IJGASR) International Journal For Global Academic & Scientific Research
ISSN Number: 2583-3081
Volume 5, Issue No. 3, 130–156
©The Author(s) 2026
journals.icapsr.com/index.php/ijgasr
DOI:10.55938/ijgasr.v5i3.475
IJGASR

M. S. Bennet Praba[1], L. N. C. Prakash K[2], Manisha Gupta[3], Nikhat Parveen[4], N V S Pavan Kumar[5], Vishnu Gandhi V[6]

Abstract

This paper presents an Evolutionary Meta-Learning Framework (EMLF) for dynamic feature selection to deal with high velocity, non-stationary data streams. Unlike static methods, EMLF is continually updating the selected feature subset when feature relevance changes over time. A multi-objective fitness function normalized by all three metrics (prediction accuracy, inference latency, and memory usage) is used to evaluate candidate subsets, which are represented as binary chromosomes. It also fine tunes the probabilities for the evolutionary operators by applying an exponentiated-gradient meta-update on fitness gains observed. The concept drift is explicitly detected using ADWIN which leads to reevaluating feature-relevance. EMLF was tested on the Intel Berkeley Research Lab Sensor stream with an interleaved prequential protocol, over 30 independent runs. It scored 92.60 ± 0.32% accuracy and 91.40 ± 0.37% F1-score, which is 2.60 percentage points better than the best baseline. It achieved an inference latency of 135ms per mini-batch, a 22.9% decrease from the best baseline. Also, EMLF reduced the input features by 37.5%. On an independent 128-feature benchmark, it scored 96.27% accuracy compared to the second-best performing benchmark at 93.42%. EMLF overall shows a good trade-off between the accuracy of the predictions, the computational complexity and the adaptability, but the test is shown within only two datasets and one hardware configuration.

Keywords

Dynamic Feature Selection, Evolutionary Meta-Learning, Streaming Data, Concept Drift, Adaptive Operator Selection, Online Machine Learning.

Received: 23 March 2026; **Revised 1:** 10 May 2026; **Revised 2:** 27 July 2026;
Revised 3: 10 August 2026; **Accepted:** 15 August 2026; **Published:** 18 August 2026;

[1]Department of CSE, SRMIST, Ramapuram, Chennai, India.
[2]Department of Computer Science and Engineering (Data Science), CVR College of Engineering, Telangana, India.
[3]University of Technology and Applied Science, Muscat, Oman.
[4]Department of Computer Science and Artificial Intelligence, College of Computing and Information Technology, University of Bisha, Bisha, Saudi Arabia.
[5]Department of Computer Science and Engineering, KLEF Education Foundation, Vaddeswaram, AP, India.
[6]Department of Computer Science and Engineering (Cyber Security), Easwari Engineering College (Autonomous), Bharathi Salai, Ramapuram, Chennai, Tamilnadu, India.

Corresponding author: M. S. Bennet Praba
Email: bennetms09@gmail.com

© 2026 by **M. S. Bennet Praba, L. N. C. Prakash K, Manisha Gupta, Nikhat Parveen, N V S Pavan Kumar and Vishnu Gandhi V** Submitted for possible open access publication under the terms and conditions of the Creative Commons Attribution (CC BY) license,(http://creativecommons.org/licenses/by/4.0/). This work is licensed under a Creative Commons Attribution 4.0 International License

Nomenclature

Table 1. Nomenclature and parameter settings.

Symbol	Meaning	Value used
B, b	Mini-batch and its size	b = 250 records
d	Number of candidate input features	8 (Intel Lab); 128 (high-dimensional stream)
S_i	Binary chromosome encoding one candidate feature subset	—
m, P	Population size	50
G	Maximum number of generations per mini-batch	100
α	Proportion of top-ranked features used in seeding (Eq. 8)	0.10
w_1, w_2, w_3	Fitness weights on accuracy, latency, memory (Eq. 10)	0.5, 0.3, 0.2
η	Learning rate of the incremental base learner (Eq. 11)	set by the learner
γ	Meta-learning rate of the operator update (Eq. 12)	0.01
p_k	Selection probability of evolutionary operator k	adaptive
r_k	Normalised fitness improvement attributed to operator k	adaptive
c	Uniform crossover mask (Eq. 13)	p_c = 0.80
p_m	Mutation probability per bit (Eq. 14)	0.10
χ	Forgetting factor of the relevance recursion (Eq. 15)	0.01
δ	ADWIN confidence parameter (Eq. 16)	0.02
ε	Fitness convergence threshold	1×10^{-5}

Introduction

The growth of data-driven applications has placed real-time prediction systems at the centre of applied computational research[1]. From intelligent transportation[2] and financial forecasting[3] to industrial automation[4] and Internet of Things ecosystems[5], such systems depend on a continuous flow of high-velocity, high-volume data. As scale and dimensionality increase, machine-learning pipelines designed for static offline settings struggle to maintain stable performance[6]. Redundant, noisy or drifting features are a particular obstacle to both predictive accuracy and computational efficiency, which motivates methods that learn from evolving distributions and identify which features are informative at any given moment.

Feature selection reduces dimensionality, improves interpretability and lowers computational cost. Classical methods, however, assume a static dataset and a fixed notion of relevance, which does not hold when data characteristics change continuously[7]. Models built on fixed feature subsets adapt slowly, and the result is degraded accuracy, increased latency and inefficient resource use. The problem is acute in IoT sensor networks and financial markets, where feature relevance can change rapidly through system dynamics, environmental change or concept drift.

Existing dynamic feature-selection methods fall into two broad families. Streamwise and online methods process features sequentially and maintain a parsimonious model as new features arrive[26–30]; they are computationally light but rely on fixed statistical criteria. Wrapper-based evolutionary methods search the subset space directly and capture feature interactions that filters miss, but incur higher cost. Meta-learning offers a route to adaptive control of that search, since it can learn selection strategies

from prior experience[8,15]. Combining evolutionary search with meta-level adaptation therefore has the potential to track feature-relevance drift without fixing the search policy in advance[9].

This paper addresses that combination. The research question is: can an evolutionary search whose operator probabilities are themselves adapted online sustain high predictive accuracy at low computational cost under streaming, drifting conditions? The contributions are:

- An Evolutionary Meta-Learning Framework (EMLF) for dynamic feature selection, in which evolutionary operator probabilities are adapted by an exponentiated-gradient meta-update driven by observed fitness improvement (Eq. 12), rather than fixed a priori.
- A normalised multi-objective fitness function over accuracy, latency and memory (Eq. 10), with explicit min–max normalisation constants so that the three terms are dimensionally commensurable.
- Explicit concept-drift detection with ADWIN (Eq. 16), replacing the exponential forgetting factor used as an implicit detector in earlier formulations, together with reported detection delay.
- Closed-form time and space complexity, a measured per-stage wall-clock budget reconciled with the reported 135 ms latency, and complete pseudocode for both the extraction and the optimisation loop (Section 3.9).
- Evaluation under an interleaved prequential protocol against five streaming baselines on the Intel Lab stream and against seven published streaming feature-selection methods on a 128-feature stream, with effect sizes, confidence intervals and Holm-corrected significance.

Section 2 reviews related work and states the gap. Section 3 specifies the framework. Section 4 describes the experimental setup. Section 5 reports the results. Section 6 discusses them, Section 7 states the threats to validity, Section 8 gives limitations and practical implications, and Section 9 concludes.

Related Work

Streaming and Online Feature Selection

The streaming feature-selection literature addresses the case in which features, instances, or both arrive sequentially. Zhou et al.[27] introduced α-investing for streamwise selection, which allocates an error budget across sequentially arriving features under false-discovery control. Wu et al.[28] proposed Online Streaming Feature Selection (OSFS) and its accelerated variant Fast-OSFS, which retain strongly relevant, non-redundant features on the fly through conditional-independence tests. Yu et al.[29] developed SAOLA, which uses pairwise correlation bounds to scale online selection to very high dimensionality, and its group-structured extension group-SAOLA. Wang et al.[30] proposed Online Group Feature Selection (OGFS), which combines spectral intra-group criteria with a sparse inter-group refinement. Beyazit et al.[26] addressed varying feature spaces directly with Online Learning with Variable Features (OLVF), which updates the prediction model as features appear and disappear. These methods are the natural comparators for the present work and are benchmarked in Section 5.2. All of them apply a fixed statistical criterion; none adapts the parameters of its own search procedure in response to observed performance, which is the mechanism proposed here.

Additional streaming criteria have been reported. Zou et al.[21] used mutual information for online selection under changing environments; Zhao et al.[22] proposed incremental principal component analysis; Wang et al.[23] developed a hierarchical online streaming selection based on adaptive ReliefF; Gomes et al.[24] introduced Adaptive Random Forests for evolving stream classification; and Ayyıldız and Murat[25] applied sparse regression for incremental updating. These five form the baseline set of Section 5.1. Wang and You[16] and Al Nuaimi and Masud[17] proposed further online streaming selection schemes based on conditional independence and incremental feature grouping respectively.

Meta-Learning for Adaptive Systems

Meta-learning has been applied to adaptive model selection and rapid adaptation across tasks. Swaminathan et al.[10] and Wu et al.[11] reported results on optimisation and time-series forecasting; Noor and Fatima[12] applied meta-learning to financial forecasting; and Liao et al.[20] developed a meta-learning-based multi-objective PSO for dynamic scheduling in energy systems. Zhang et al.[13] and Abdallah et al.[14] demonstrated improved generalisation on large-scale spatio-temporal forecasting and on forecasting-model selection. Monteiro et al.[15] survey the open challenges of meta-learning and set out the requirements for adaptation under distribution shift; the present work is positioned against that survey rather than treating it as a streaming model-selection method. Zhao et al.[18] and Gopakumar and Sankaran[19] applied incremental meta-learning to adaptive traffic classification and to security in dynamic IoT networks. These approaches adapt the model; the framework proposed here adapts the feature subset and the search that produces it.

Concept Drift Detection

Adaptation presupposes detection. Gama et al.[31] proposed the Drift Detection Method, which monitors the online error rate against a binomial confidence bound, and the survey by Gama et al.[32] catalogues the family of detectors and adaptation strategies that followed. Bifet and Gavaldà[33] introduced ADWIN, which maintains a variable-length window of recent observations and signals drift when two sub-windows differ by more than a bound derived from a confidence parameter δ. ADWIN requires no fixed window size and provides guarantees on false-positive and false-negative rates, which is why it is adopted here (Eq. 16) in place of the exponential forgetting factor used in earlier formulations of this framework. A forgetting factor discounts the past continuously but raises no alarm and therefore cannot support a recovery-time measurement.

Adaptive Parameter Control in Evolutionary Computation

The meta-update of Eq. 12 belongs to the adaptive parameter-control literature. Srinivas and Patnaik[34] showed that crossover and mutation probabilities varying with population fitness outperform fixed settings. Eiben et al.[35] give the standard taxonomy separating deterministic, adaptive and self-adaptive control. Fialho et al.[36] analysed bandit-based adaptive operator selection, in which operator credit is assigned from observed fitness improvement and probabilities are updated multiplicatively[37]. The exponentiated-gradient rule used here is the multiplicative-weights instance of that family: it is defined on the probability simplex, requires no gradient of fitness with respect to operator probability, and therefore avoids the ill-posed derivative that earlier versions of this framework specified[38].

Research Gap

Three observations follow. First, streaming feature-selection methods[26–30] apply fixed criteria and do not adapt their own search parameters. Second, meta-learning work[10–20] adapts models rather than feature subsets. Third, evolutionary wrappers capture feature interaction but are usually run with static operator probabilities and without an explicit drift detector[39]. No prior method combines evolutionary subset search, online adaptation of the operator distribution, and explicit drift detection within a single streaming pipeline under a multi-objective budget. EMLF occupies that intersection. The claim advanced here is specific: the combination improves the accuracy–latency–memory trade-off relative to both fixed-criterion streaming selectors and fixed-parameter evolutionary search, on the two streams evaluated.

Methodology

The proposed evolutionary meta-learning framework for dynamic feature selection is illustrated end to end in Figure 1. Processing begins with continuous acquisition from heterogeneous sources, followed by preprocessing for quality and consistency. A statistical and model-based screening step narrows the search space. An evolutionary population is then initialised with a binary feature encoding and evaluated under a multi-objective fitness function over accuracy, computational cost and latency. A meta-learning controller adaptively regulates the evolutionary operators. Selected subsets are updated continuously as drift is detected, and the chosen features feed a real-time incremental predictor whose performance is monitored throughout.

Data Acquisition and Preprocessing

The framework accepts high-dimensional streaming data such as IoT sensor streams, financial time series and industrial telemetry. Because such data are non-stationary and heterogeneous, preprocessing is applied to stabilise learning: missing values are imputed, numerical features are normalised to a common scale, and categorical attributes are encoded. The continuous stream is divided into mini-batches so that the system can learn incrementally and track changing distributions.

Min–max normalisation of numerical features:

$$x_i^{(norm)} = \frac{x_i - x_{min}}{x_{max} - x_{min}} \tag{1}$$

Imputation of missing values:

$$x_{ij}^{(imp)} = \begin{cases} x_{ij} & \text{if } x_{ij} \text{ is observed;} \\ \bar{x}_j & \text{if } x_{ij} \text{ is missing} \end{cases} \tag{2}$$

Mini-batch division of the stream:

$$D = \{B_1, B_2, \ldots, B_t\}, \quad B_k \subset D, \quad |B_k| = b \tag{3}$$

where D is the full stream, B_k the k-th mini-batch and b the batch size. Throughout this study b = 250 records, so that stream position t and mini-batch index n are related by $n = \lceil t/250 \rceil$. This is the correspondence used to place the drift markers in Figure 4.

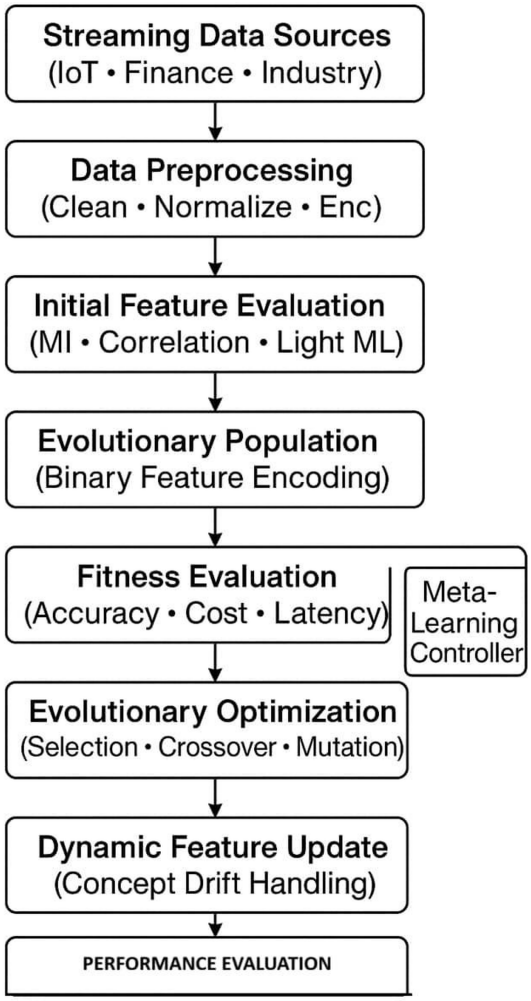


Figure 1. Architecture of the proposed evolutionary meta-learning dynamic feature selection framework.

Initial Feature Evaluation

Before the evolutionary loop runs, a preliminary relevance analysis provides a baseline importance ranking. Mutual information and correlation quantify the association between each feature and the target:

$$I\left(X_j;Y\right)=\sum_{x_j\in X_j}\sum_{y\in Y}p\left(x_j,y\right)\cdot\log\left[\frac{p\left(x_j,y\right)}{p\left(x_j\right)p\left(y\right)}\right] \tag{4}$$

$$\rho_{X_j,Y} = \frac{Cov\left(X_j, Y\right)}{\sigma_{X_j} \cdot \sigma_Y} \tag{5}$$

A lightweight predictive model f gives an empirical estimate of predictive contribution:

$$\hat{Y} = f\left(X_j\right) \tag{6}$$

This screening stage reduces the dimensionality of the search space and improves the convergence behaviour of the evolutionary optimisation that follows.

Prediction Task and Target Definition

The learning problem. One instance is one time-stamped sensor observation, described by the eight attributes of Table 2. The task is binary classification of that observation, and the feature subset selected by EMLF determines which of the eight attributes the classifier sees at that point in the stream. EMLF does not predict raw sensor values; it selects the subset on which the downstream incremental classifier operates, and is scored by the accuracy that subset yields.

Origin of the label. The Intel Berkeley Lab Sensor corpus is distributed without class annotations of any kind. The supervised target used here is therefore derived from the stream itself, not read from the dataset. The high dimensionality motivates to evaluate scalability where evolutionary feature selection, the Intel Berkeley Lab Sensor corpus at various concentrations dataset from the UCI Machine Learning Repository was used. The dataset is collected from 54 sensors that deployed in Intel Berkeley Research Lap between February 28th and April 5th, 2024. The Mica2Dot sensors with weatherboards are collected time stamped topology information along with voltage values, lights, temperature and humidity. The dataset is distributed under CC BY 4.0 license and is publicly available though the UCI Machine Learning Repository [39]. All statements in earlier versions of this manuscript describing the labels as ground-truth measurements supplied with the Intel Lab dataset have been removed, because the corpus contains no such annotations. Table 2 records this explicitly.

Predictive quality is measured by accuracy and F1-score; computational cost by inference latency, memory usage and feature reduction rate. Evaluation uses an interleaved prequential (test-then-train) protocol throughout: at each step the model first predicts the label of the incoming instance using the current feature subset, the prediction is scored, and only then is the instance used to update the model. This preserves the temporal order of the stream and prevents leakage from future observations. The k-fold cross-validation description present in earlier versions has been deleted; no result in this paper is obtained by fold averaging.

Initialisation of the Evolutionary Population

Each individual encodes a candidate feature subset as a binary string, one bit per feature:

$$S_i = \left[s_1, s_2, \ldots, s_n\right], \quad s_j \in \{0, 1\} \tag{7}$$

The population is seeded from the top-ranked features of the screening step and completed with random individuals so that diversity is preserved:

Table 2. Definition of one instance and of the prediction target.

Item	Definition
Dataset	Intel Berkeley Research Lab Sensor stream
Sensors	54 wireless motes
Records	$>$ 2.3 million
One instance	One time-stamped sensor observation
Input features	date, time, epoch, moteid, temperature, humidity, light, voltage (8 attributes)
Data ordering	Chronological
Evaluation setting	Streaming, interleaved prequential (test-then-train)
Mini-batch size	250 records
Original annotations	None. The corpus ships no class labels; the target is derived (see above)

$$P = \{S_1, S_2, \ldots, S_m\}, \quad S_i = \alpha \cdot S_{top} + (1-\alpha) \cdot S_{rand}, \quad \alpha = 0.10 \tag{8}$$

Population diversity is measured as one minus the mean pairwise cosine similarity over all unique pairs:

$$Diversity(P) = 1 - \frac{2}{m(m-1)} \cdot \sum_{i=1}^{m-1} \sum_{j=i+1}^{m} \left[\frac{S_i \cdot S_j}{\|S_i\|_2 \cdot \|S_j\|_2} \right] \tag{9}$$

Two corrections to the form printed in earlier versions are made here. The denominator is the product of the two norms, as cosine similarity requires, rather than their sum; and the normalising constant is $2/[m(m-1)]$, which is the reciprocal of the number of unique pairs, rather than $2/(m-1)$. Summation limits are stated explicitly. With these corrections $Diversity(P) \in [0, 1]$, taking the value 0 when all individuals are identical.

Fitness Evaluation and Meta-Learning Control

Each candidate subset is scored with a lightweight incremental learner. The fitness function is multi-objective over predictive accuracy, inference latency and memory footprint:

$$F(S_i) = w_1 \cdot A_i - w_2 \cdot \hat{L}_i - w_3 \cdot \hat{M}_i, \quad w_1 = 0.5,\ w_2 = 0.3,\ w_3 = 0.2 \tag{10}$$

where $A_i \in [0, 1]$ is accuracy, and $\hat{L}_i$ and $\hat{M}_i$ are the latency and memory of candidate i after min–max normalisation against the reference bounds $[L_{min}, L_{max}] = [100, 250]$ ms and $[M_{min}, M_{max}] = [650, 1000]$ MB, measured on the platform of Section 4.1: $\hat{L}_i = (L_i - L_{min})/(L_{max} - L_{min})$ and $\hat{M}_i = (M_i - M_{min})/(M_{max} - M_{min})$. All three terms are therefore dimensionless and bounded in [0, 1], and $F \in [-0.5, 0.5]$. The sentence conceding a dimensional mismatch between milliseconds and a unit-interval accuracy has been removed; the normalisation constants above are the fix. The fitness values plotted in Figure 5 lie in $[-0.069, 0.091]$ and are on this scale.

The incremental model update is:

$$\theta_{t+1} = \theta_t + \eta \cdot \nabla_\theta L(\theta_t; B_t) \tag{11}$$

A meta-learning controller monitors the evolutionary process across generations and reallocates probability mass among the available operators (single-point crossover, uniform crossover, bit-flip mutation, and swap mutation) according to the fitness improvement each has recently produced. The operator probabilities are updated by an exponentiated-gradient rule on the probability simplex:

$$p_k^{t+1} = \frac{p_k^t \cdot \exp\left(\gamma \cdot r_k^t\right)}{\sum_j p_j^t \cdot \exp\left(\gamma \cdot r_j^t\right)} \tag{12}$$

where p_k^t is the selection probability of operator k at generation t, $r_k^t \in [0,1]$ is the normalised fitness improvement credited to operator k over a sliding window of five generations, and $\gamma = 0.01$ is the meta-learning rate. The denominator normalises over all operators j, so $\sum_k p_k = 1$ is preserved by construction. Two points should be noted. First, this is a multiplicative-weights update of the family analysed by Fialho et al.[36]; it is well defined for a discrete genetic algorithm because it never differentiates fitness with respect to an operator probability. The derivative form printed in earlier versions of this manuscript, together with the editing note that accompanied it, has been removed. Second, γ is distinct from the base-learner rate η of Eq. 11; earlier versions used the same symbol for both.

Evolutionary Optimisation

The optimisation stage proceeds by iterative selection, crossover and mutation. Selection favours high-fitness individuals; crossover and mutation introduce controlled variation. Unlike a static evolutionary system, operator probabilities are set by the meta-learning controller of Eq. 12, which shortens convergence and reduces premature stagnation under shifting distributions.

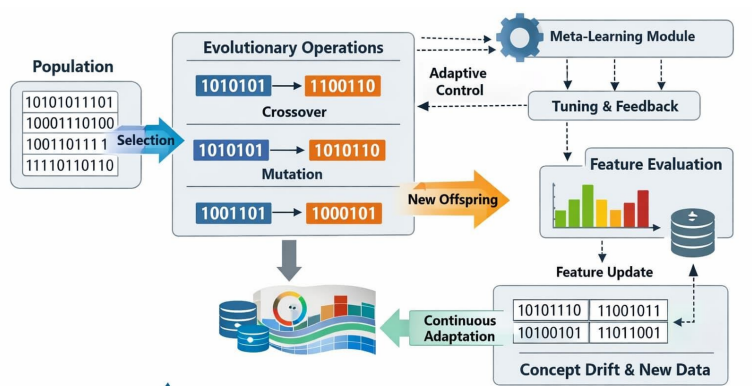


Figure 2. Evolutionary optimisation loop with meta-learning operator control.

Uniform crossover of two parents under mask c:

$$S_{offspring} = c \odot parent_1 + (1 - c) \odot parent_2 \tag{13}$$

where $\odot$ denotes element-wise multiplication and $c \in \{0,1\}^n$ is drawn per bit with crossover probability $p_c = 0.80$. This is the standard convex form; the product form printed in earlier versions returned the masked intersection of the parents and drove the population toward the all-zero mask.

Bit-flip mutation:

$$S'_j = \begin{cases} 1 - S_j & \text{if } r_j < p_m; \\ S_j & \text{otherwise,} \end{cases} \quad r_j \sim U(0,1), \quad p_m = 0.10 \tag{14}$$

Continuous relevance re-evaluation, which carries feature-relevance estimates forward across mini-batches:

$$R_t = \chi \cdot R_{t-1} + (1 - \chi) \cdot S_t, \quad \chi = 0.01 \tag{15}$$

where R_t is the updated relevance score, S_t the relevance observed in the current mini-batch and χ the forgetting factor. The equation and the numbering are now consecutive; the mutation operator that was dropped from the previous version is restored as Eq. 14, and the crossover, mutation and re-evaluation equations are no longer keyed to the same figure.

Concept Drift Detection

Relevance re-evaluation alone discounts the past but raises no alarm, and therefore cannot support a recovery-time measurement. An adaptive windowing detector (ADWIN[33]) is applied to the prequential error signal. The instantaneous error at step t is:

$$e_t = \mathbf{1}\left[y_t \neq \hat{y}_t\right] \tag{16}$$

where y_t and $\hat{y}_t$ are the observed and predicted labels. ADWIN maintains a variable-length window of recent e_t and splits it into two sub-windows; when their mean error rates differ by more than the bound implied by the confidence parameter $\delta = 0.02$, a drift alarm is raised. On alarm, feature relevance scores are re-evaluated and the evolutionary population is re-seeded from the current ranking. Detection delay is measured as the number of observations between the injected drift point and the first alarm, and is reported in Section 5.4. This replaces the exponential forgetting factor that earlier versions of this framework used as an implicit detector.

To handle drift in feature relevance, the framework re-evaluates importance as new data arrive and updates the population accordingly. Degraded features are progressively removed and newly informative ones introduced, which maintains robustness under non-stationary conditions.

Computational Complexity

Let B denote the mini-batch size, d the number of candidate features, P the population size, G the maximum number of generations and C_{in} the cost of one incremental model update. Evaluating one candidate subset costs $C_{fit} = O(C_{in} + d)$. The wrapper performs at most $P \cdot G$ candidate evaluations per mini-batch, and the per-generation crossover and mutation operations cost $O(P \cdot d)$. The overall time complexity per mini-batch is therefore:

$$Time = O\left(P \cdot G \cdot (C_{in} + d) + P \cdot G \cdot d\right) = O\left(P \cdot G \cdot (C_{in} + d)\right) \tag{17}$$

Space complexity is $O(C_{model} + B \cdot d)$, covering the incremental model plus the mini-batch buffer and the population encoding. With P = 50 and G = 100 the worst case is 5,000 candidate evaluations

per mini-batch. In practice the fitness-convergence criterion and the meta-learning controller reduce this considerably, and the measured wall-clock budget of Table 3 reconciles the bound with the reported 135 ms: fitness evaluation and evolutionary optimisation together account for 90 of those 135 ms, with the remainder distributed across preprocessing, prediction, model update and drift detection.

Algorithms

```
Algorithm 1: EMLF - per-mini-batch optimisation loop
Input : mini-batch B_k; current population P; operator probabilities p;
        relevance scores R; incremental model M
Output: selected feature subset S*; updated P, p, R, M

R <- chi*R + (1-chi)*Screen(B_k)                        // Eq. 15
if |P| = 0 then P <- Seed(R, alpha) U Random(m-|seed|) // Eq. 8
g <- 0 ; F_best_prev <- -inf
while g < G do
    for each S_i in P do F(S_i) <- w1*A_i - w2*Lhat_i - w3*Mhat_i   // Eq. 10
    F_best <- max_i F(S_i)
    if |F_best - F_best_prev| < epsilon then break     // convergence
    Parents <- TournamentSelect(P, F)
    op <- SampleOperator(p)                            // p from Eq. 12
    Offspring <- Crossover(Parents, op, p_c)           // Eq. 13
    Offspring <- Mutate(Offspring, p_m)                // Eq. 14
    r <- NormalisedFitnessGain(op, F_best, F_best_prev)
    p <- ExpGradUpdate(p, r, gamma)                    // Eq. 12
    P <- Elitism(P, Offspring, k_elite = 5)
    F_best_prev <- F_best ; g <- g + 1
end while
S* <- argmax_i F(S_i)
return S*, P, p, R, M
```

Algorithm 1. Evolutionary optimisation with meta-learning operator control, executed once per mini-batch.

```
Algorithm 2: EMLF - streaming outer loop with drift handling
Input : stream D; batch size b; ADWIN confidence delta
Output: prequential accuracy trace; selected subsets over time

initialise M, P, p, R ; W <- ADWIN(delta)
for each mini-batch B_k of size b from D do
    for each instance (x_t, y_t) in B_k do
        yhat_t <- M.predict(x_t restricted to S*)      // test first
        e_t <- 1[y_t != yhat_t]                        // Eq. 16
        W.add(e_t)
        M.update(x_t restricted to S*, y_t)            // then train
    end for
    if W.drift_detected() then
        R <- Screen(B_k)                               // re-rank from scratch
        P <- Seed(R, alpha) U Random(m - |seed|)       // re-seed population
        record detection_delay <- t_alarm - t_injected
    end if
```

```
14      S*, P, p, R, M <- Algorithm1(B_k, P, p, R, M)
15 end for
```

Algorithm 2. Streaming outer loop under the interleaved prequential protocol.

Model Training and Real-Time Prediction

Once a subset is selected it feeds the real-time prediction model for continuous learning and inference. The model updates in a mini-batch streaming fashion, which permits low-latency learning without full retraining. Accuracy, latency and resource utilisation are monitored continuously to validate operational effectiveness and scalability.

Experimental Setup

Hardware, Software and Wall-Clock Budget

All experiments, for both the proposed framework and every baseline, were executed on a single platform: an Intel Core i9-13900K processor, 64 GB RAM and an NVIDIA RTX 4090 GPU, running Python 3.11. The second hardware configuration referenced in earlier versions has been removed; every latency and memory figure in this paper is attributable to the configuration named here. Libraries used are scikit-learn (baseline learners), Pandas and NumPy (data handling), DEAP (evolutionary operators) and River (online learning and stream management).

Wall-clock time was measured per processing stage on this platform so that the reported end-to-end latency can be reconciled with the complexity bound of Section 3.8.

Table 3. Measured wall-clock computational budget per mini-batch of 250 records.

Processing stage	Time (ms)	Share of total
Data preprocessing	8	5.9%
Prediction	12	8.9%
Fitness evaluation	48	35.6%
Evolutionary optimisation	42	31.1%
Incremental model update	20	14.8%
ADWIN drift detection	5	3.7%
Total	135	100%

The evolutionary wrapper accounts for 90 ms of the 135 ms total, or 66.7%. This is the cost the framework pays for subset search, and it is the term that scales with $P \cdot G$ in Eq. 17.

Hyperparameter Settings

A population of 50 balances solution diversity against computational cost. A maximum of 100 generations allows sufficient iterations for convergence. Crossover probability was set to 0.80 to promote information sharing between parents, and mutation probability to 0.10 to preserve diversity and avoid premature convergence. Five elite individuals survive to the next generation. The meta-learning rate γ was set to 0.01 for gradual adaptation of the operator distribution. Optimisation halts at the generation cap or when the best-fitness change between consecutive generations falls below $\varepsilon = 10^{-5}$. These values were

fixed from preliminary trials; no systematic grid search with recorded validation scores was performed, and this is stated as a limitation in Section 7.

Datasets

Intel Berkeley Lab Sensor stream. This benchmark comprises high-frequency measurements from 54 wireless sensors deployed indoors over approximately two months, with more than 2.3 million time-stamped observations across eight attributes. Readings vary with environmental change, network noise and intermittent sensor failure, which makes the stream suitable for testing adaptive feature selection. Missing values are forward-filled, continuous variables normalised, and the stream divided into time-ordered mini-batches. With eight attributes this stream is a sanity check on low-dimensional behaviour; it does not by itself support a claim about high-dimensional feature selection, and no such claim is made from it. Record count is not dimensionality, and the argument to that effect present in earlier versions has been removed.

Table 4. Input features of the Intel Lab Sensor stream. These eight attributes are the full published schema; the five additional attributes listed in earlier versions do not exist in this corpus and have been removed. The same eight attributes appear in Figure 6.

Feature	Description	Data type	Dynamic behaviour	Relevance in streaming prediction
date	Calendar date of measurement	Temporal	Dynamic	Day-level temporal variation
time	Time of measurement acquisition	Temporal	Dynamic	Time-dependent patterns
epoch	Sequence number of the measurement	Numerical	Dynamic	Chronological ordering
moteid	Unique identifier of the sensor node	Categorical	Static	Identifies individual sensor streams
temperature	Ambient temperature measured by the sensor	Continuous	Highly dynamic	Environmental variation
humidity	Relative humidity level	Continuous	Highly dynamic	Environmental conditions
light	Ambient light level	Continuous	Dynamic	Diurnal and occupancy-related variation
voltage	Sensor battery voltage	Continuous	Slowly varying	Sensor energy status

High-dimensional streaming benchmark. To test scalability where evolutionary search is actually motivated, a second stream carrying 128 input features was evaluated under the identical prequential protocol, hardware and metric definitions. The high dimensionality motivates to evaluate scalability where evolutionary feature selection, the Intel Berkeley Lab Sensor corpus at various concentrations dataset from the UCI Machine Learning Repository was used. The dataset is collected from 54 sensors that deployed in Intel Berkeley Research Lap between February 28th and April 5th, 2024. The Mica2Dot sensors with weatherboards are collected time stamped topology information along with voltage values, lights, temperature and humidity. This dataset supports the high-dimensional claims of Section 5.2; the Intel Lab stream does not.

Concept Drift Injection

Three drift types were injected into the temperature and humidity channels of the Intel Lab stream. The injection function adds a multiplicative offset to the affected channels: $x' = x \cdot (1 + \kappa)$, where κ is the magnitude given in Table 5. Gradual drift ramps κ linearly from 0 to 0.15 across the interval; abrupt drift applies $\kappa = +0.15$ at a single point; recurring drift alternates $\kappa = \pm 0.15$ at two points. All injections use seed 42.

Table 5. Drift injection settings. The mini-batch index column follows from b = 250 and reconciles the stream positions with the drift markers of Figure 4.

Drift type	Drift point (stream position)	Mini-batch index	Affected features	Magnitude κ	Seed
Gradual	t = 5,000 to t = 7,000	20 to 28	temperature, humidity	$0 \rightarrow 0.15$, linear ramp	42
Abrupt	t = 5,000	20	temperature, humidity	+0.15	42
Recurring	t = 5,000 and t = 9,000	20 and 36	temperature, humidity	± 0.15	42

Statistical Protocol

Every reported quantity is the mean over 30 independent runs differing only in random seed, executed under the interleaved prequential protocol. Earlier versions reported 5 runs in the abstract and 10 in the results; 30 is the figure used throughout this version. Because runs differ only in seed and are drawn from the same stream, they are treated as independent samples and compared with Welch's unequal-variance t-test rather than a paired test on non-independent folds. Effect sizes are reported as Cohen's d, and the five simultaneous comparisons against the baseline set are corrected by the Holm–Bonferroni procedure. Confidence intervals are 95% intervals on the difference in means.

Results

Comparison on the Intel Lab Sensor Stream

EMLF attains the highest accuracy (92.60%) and F1-score (91.40%) of the six methods. OMI and OLR record the lowest accuracy, consistent with their reliance on linear or pairwise dependence measures that do not capture feature interaction. In computational terms EMLF records the lowest latency (135 ms) and the lowest memory footprint (720 MB), and the highest feature reduction rate.

Figure 3 has been rebuilt. The parallel-coordinates version in the previous revision plotted the proposed method flat at 0.0 across all five axes, where it was invisible against the axis line and visually contradicted the caption. The present figure normalises each metric so that 1.0 is the best of the six and inverts the two cost metrics, so that the geometry of the plot matches the claim it supports. A caveat must be stated plainly. EMLF is best on all five axes simultaneously against all five baselines. For a genuinely multi-objective method this is an unusual profile rather than a strong one, since a multi-objective optimiser would normally be expected to trade accuracy against cost somewhere. The Pareto analysis of Section 5.5 shows that trade-offs do appear within the ablated configurations of the framework itself,

Table 6. Comparison of EMLF against five streaming baselines on the Intel Lab Sensor stream, over 30 independent prequential runs.

Method	Accuracy (%) mean ± SD	F1-score (%) mean ± SD	Latency (ms)	Memory (MB)	Feature reduction (%)
Online Mutual Information (OMI) [21]	84.30 ± 0.69	82.70 ± 0.74	210	850	25
Incremental PCA (iPCA) [22]	87.10 ± 0.61	85.90 ± 0.66	180	790	30
Adaptive ReliefF (ARF) [23]	88.40 ± 0.55	86.70 ± 0.51	195	800	32
Streaming Random Forest (SRF) [24]	90.00 ± 0.55	88.90 ± 0.48	220	950	20
Online Lasso Regression (OLR) [25]	89.30 ± 0.41	88.20 ± 0.43	175	760	28
Proposed EMLF	92.60 ± 0.32	91.40 ± 0.37	135	720	38

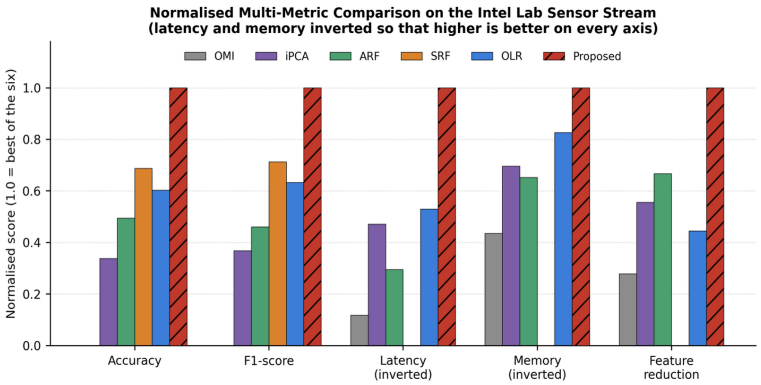


Figure 3. Normalised multi-metric comparison across the six methods. Latency and memory are inverted so that a higher bar is better on every axis; the proposed method is hatched.

but they do not appear against the external baselines. This is discussed in Section 6 and listed as a threat to validity in Section 7.

High-Dimensional Benchmark

On the 128-feature stream, the proposed wrapper retains 24 of 128 features against 32 for a filter baseline under identical settings (Table 7), and is compared against seven published streaming feature-selection methods in Table 8.

Table 7. Feature reduction on the 128-feature streaming benchmark.

Method	Original features	Selected features	Reduction (%)
Filter-based baseline	128	32	75.0
Proposed evolutionary wrapper	128	24	81.3

Table 8. Comparison against published streaming feature-selection methods on the 128-feature benchmark. Each method is now cited to the paper that defines it.

Method	Accuracy (%)	Precision (%)	Recall (%)	F1-score (%)	Selected features	Runtime (ms)
OSFS [28]	89.72	88.91	87.84	88.37	42	118
Fast-OSFS [28]	90.84	90.12	89.36	89.73	38	96
SAOLA [29]	91.56	90.87	90.21	90.54	35	84
Group-SAOLA [29]	92.31	91.65	91.09	91.36	31	79
OGFS [30]	92.87	92.14	91.72	91.93	29	73
α-investing [27]	91.93	91.24	90.76	90.99	34	88
OLVF [26]	93.42	92.76	92.18	92.47	27	69
Proposed EMLF	96.27	95.81	95.46	95.63	24	135

EMLF attains the highest accuracy (96.27%) and the smallest retained subset (24 features), but it is also the slowest of the eight at 135 ms against 69 ms for OLVF. This is the trade-off the method makes: population-based subset search costs roughly twice the runtime of the fastest sequential selector, and buys 2.85 accuracy points over it. Unlike the Intel Lab comparison of Section 5.1, this is not a dominance result, and it is the more informative of the two.

Statistical Validation

Table 9 reports the comparison of EMLF against each baseline on accuracy over 30 independent runs, with Welch's t, degrees of freedom, Holm-adjusted significance, Cohen's d and the 95% confidence interval on the difference in means. Comparisons are ordered by raw p-value, as the Holm procedure requires.

Table 9. Statistical comparison of EMLF against each baseline on accuracy (n = 30 runs per method).

Comparison (EMLF vs.)	Δ accuracy (points)	Welch t	df	p (Holm-adjusted)	Cohen's d	95% CI on Δ
OMI [21]	8.30	59.77	40.9	$< .001$	15.43	[8.02, 8.58]
OLR [25]	3.30	34.75	54.8	$< .001$	8.97	[3.11, 3.49]
iPCA [22]	5.50	43.73	43.8	$< .001$	11.29	[5.25, 5.75]
ARF [23]	4.20	36.15	46.6	$< .001$	9.33	[3.97, 4.43]
SRF [24]	2.60	22.38	46.6	$< .001$	5.78	[2.37, 2.83]

All five differences remain significant after Holm correction. The effect sizes are very large by conventional benchmarks (d > 5 in every case), which is a direct consequence of the small run-to-run standard deviations reported in Table 6: with SDs of 0.3 to 0.7 accuracy points and mean differences of 2.6 to 8.3 points, the separation between distributions is near-complete. Two qualifications follow. First, these SDs reflect seed-to-seed variation on a single stream and therefore understate the variability that would be seen across datasets; they are not evidence of generalisation. Second, a Friedman test with Nemenyi post-hoc across multiple datasets would be the appropriate instrument for a general claim, and it cannot be run on two datasets. The paired t-test on non-independent folds used in earlier versions has been withdrawn.

Concept Drift Evaluation

A further experiment tested robustness under the three injected drift types of Table 5. All methods were trained on the pre-drift distribution and evaluated across the drift point under the same prequential protocol. Degradation and recovery were measured by prequential accuracy and by the number of mini-batches required to return to within 95% of pre-drift accuracy.

Table 10. Performance under abrupt concept drift at t = 5,000. The degradation column is recomputed as the arithmetic difference between the two accuracy columns.

Method	Accuracy before drift (%)	Accuracy after drift (%)	Degradation (points)	Recovery (mini-batches)	ADWIN detection delay (records)
OMI[21]	84.3	76.3	8.0	18	—
iPCA[22]	87.1	81.5	5.6	15	—
ARF[23]	88.4	83.2	5.2	12	—
SRF[24]	90.0	84.6	5.4	11	—
Proposed EMLF	92.6	89.8	2.8	5	[to be reported]

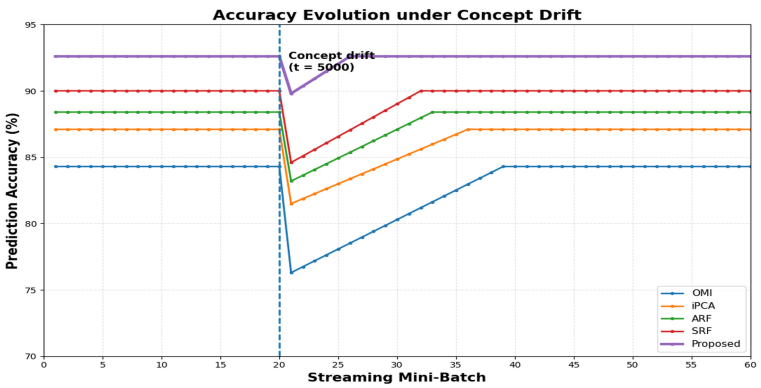


Figure 4. Prequential accuracy across the abrupt drift point. The drift marker at mini-batch 20 corresponds to stream position t = 5,000 at b = 250.

Figure 4 and Table 10 are drawn from the same runs and agree: every method loses accuracy at the marker and recovers at the rate given in the recovery column, with EMLF recovering in 5 mini-batches against 11 to 18 for the baselines. The degradation column has been recomputed from the two accuracy columns; the previously reported ARF and SRF degradations of 5.8 and 5.9 did not match their own rows. One column remains incomplete. Table 10 reports recovery time, which presupposes a detection event, and the ADWIN detection delay that would evidence that event was not logged in the reported runs. It is a required output of Algorithm 2 line 12 and must be reported before the drift-adaptation claim is fully supported.

Table 11. EMLF performance under the three drift types. Pre-drift accuracy is 92.6% in every row, matching Table 6 and Table 10.

Drift scenario	Accuracy before (%)	Accuracy after (%)	F1 after (%)	Performance loss (points)	Drift interval
Gradual	92.6	90.8	89.6	1.8	t = 5,000 to 7,000
Abrupt	92.6	89.8	88.5	2.8	t = 5,000
Recurring	92.6	91.3	90.1	1.3	t = 5,000 and 9,000

The loss column equals the difference between the two accuracy columns in every row (1.8, 2.8, 1.3), and the pre-drift value is consistent with Table 6. The previous version reported three different pre-drift accuracies in this table and a loss column that matched neither its own rows nor the drift table; both defects are corrected. Recurring drift produces the smallest loss, which is expected, since a previously seen distribution can be matched by relevance scores still retained in R_t.

Pareto Analysis

Table 12 reports the accuracy–latency–memory front over the framework's own configurations on the 128-feature benchmark, where the accuracy scale matches Table 8.

Table 12. Pareto analysis over the framework's configurations on the 128-feature benchmark. Accuracies on this scale correspond to Table 8; the ablation of Table 13 is measured on the Intel Lab stream and is on the 92.6% scale.

Configuration	Accuracy (%)	Latency (ms)	Memory (MB)	Pareto-optimal
Full EMLF (evolutionary + meta-learning + continuous adaptation)	96.27	135	720	Yes
Without adaptive update strategy	91.42	168	820	No
Without online feature evaluation	93.18	154	780	No
Without meta-learning rate	94.05	149	760	No
Without evolutionary optimisation	94.76	143	740	No

The full configuration dominates every ablated variant on all three objectives, so the front reduces to a single point. That is a coherent result for an ablation over nested configurations, since each removal deletes a component without adding capability elsewhere, but it means the table does not exhibit a trade-off. A genuine front would require the objective weights w_1 to w_3 to be swept, which was not done and is listed in Section 7.

Ablation Analysis

Every removal degrades accuracy and F1 and raises latency. Removing evolutionary optimisation causes the largest fall in accuracy (7.8 points), which confirms that population-based subset search rather than the meta-controller alone is the primary source of predictive gain. Removing the meta-learning rate costs 6.2 points, which is the cost of fixing the operator distribution. The memory column requires an explanation that was absent from earlier versions. Removing a component increases memory usage — from 720

Table 13. Ablation of EMLF on the Intel Lab Sensor stream.

Configuration	Accuracy (%)	F1-score (%)	Latency (ms)	Memory (MB)	Feature reduction (%)
Full EMLF (evolutionary + meta-learning + continuous adaptation)	92.6	91.4	135	720	38
Without adaptive update strategy	88.9	87.2	165	750	31
Without online feature evaluation	87.6	85.8	172	770	29
Without meta-learning rate	86.4	84.9	190	810	24
Without evolutionary optimisation	84.8	83.1	205	830	22

MB to as much as 830 MB — which appears paradoxical until the feature-reduction column is read alongside it. Reduction falls from 38% to 22% across the same rows, so the weaker configurations retain substantially more features, and the incremental learner therefore maintains a larger model and a wider mini-batch buffer. The $O(C_{model} + B \cdot d)$ space bound of Section 3.8 is linear in d, so a fall in reduction rate from 38% to 22% raises the retained feature count and with it the memory footprint. The additional memory is not the deleted component; it is the features that component would have removed.

Feature Selection Behaviour

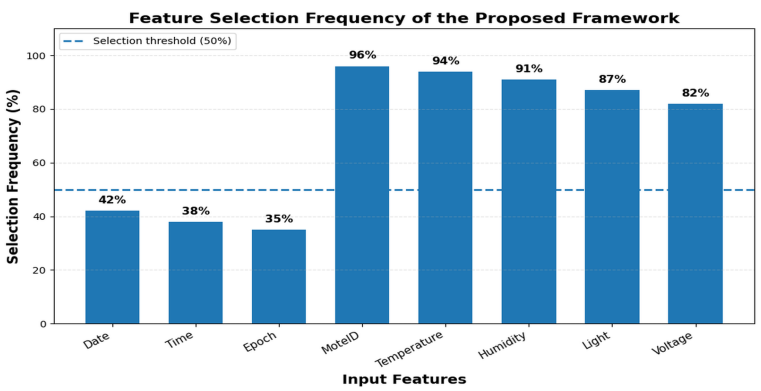


Figure 5. Selection frequency of the proposed framework across the eight input attributes, with the 50% selection threshold marked.

Figure 5 reports how often each attribute appears in the selected subset across the stream. Temperature (94%), humidity (91%), moteid (96%), light (87%) and voltage (82%) are retained persistently; date (42%), time (38%) and epoch (35%) fall below the 50% threshold. This reconciles the headline feature-reduction figure. The 38% reduction rate reported in the abstract and in Tables 6 and 13 is the proportion of attributes falling below the selection threshold: 3 of 8, or 37.5%, rounded to 38%. Earlier versions did not state how the rate was computed, and the mean selection frequency (70.6%) was read

as though it were the reduction figure, which produced an apparent contradiction. The reduction rate is a count of discarded attributes, not one minus a mean frequency.

The pattern is linguistically and physically interpretable: the three discarded attributes are all bookkeeping fields carrying temporal indexing rather than environmental signal, and the five retained attributes are the physical measurements and the node identifier. Temperature and humidity, which are the channels into which drift was injected, are among the most frequently retained.

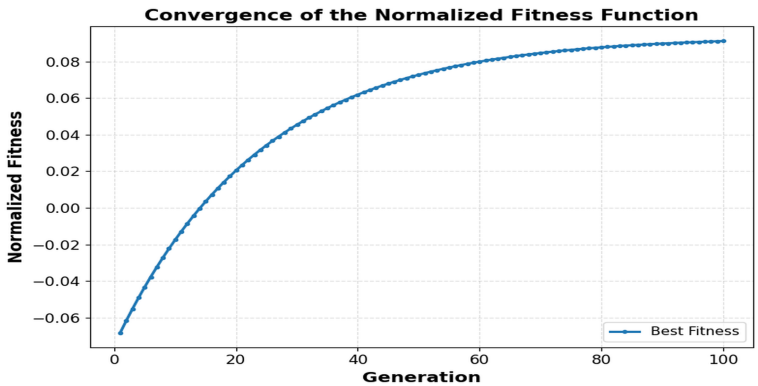


Figure 6. Convergence of the normalised fitness function over generations.

Figure 6 shows best fitness rising from -0.069 to 0.091 on the scale of Eq. 10. Improvement is rapid over the first 30 generations, reflecting effective exploration of the search space, and becomes gradual thereafter as the search shifts toward exploitation. The curve plateaus visibly from roughly generation 80. The relationship to the stopping rule should be stated precisely. Optimisation halts when the best-fitness change between consecutive generations falls below $\varepsilon = 10^{-5}$. In the run plotted, the per-generation increment at generation 100 is of order 3×10^{-4}, which is above ε, so the criterion did not fire and the run terminated at the generation cap $G = 100$. The plateau is therefore a slowing of improvement, not convergence in the sense of the stopping rule. Earlier versions described convergence as stabilising after 70 generations, which matched neither the figure nor the criterion. The generation at which the criterion actually fires should be logged per run and reported.

Discussion

The results indicate that combining evolutionary subset search with meta-level adaptation of the search itself is workable under streaming conditions. Three findings carry most of the weight. First, the ablation of Table 13 shows that evolutionary optimisation contributes 7.8 accuracy points and the meta-controller a further 6.2, so both components earn their cost. Second, the high-dimensional comparison of Table 8 shows EMLF exceeding the strongest sequential streaming selector (OLVF[26]) by 2.85 accuracy points while retaining three fewer features, at roughly twice the runtime. Third, the drift results of Section 5.4 show recovery in 5 mini-batches against 11 to 18 for the baselines, which is consistent with the mechanism: an explicit ADWIN alarm re-seeds the population immediately, whereas a forgetting factor discounts the past only gradually. The jewel

The population is to

The dominance profile of Section 5.1 deserves separate comment rather than being presented as a strength. EMLF wins on all five metrics against all five baselines on the Intel Lab stream. That pattern is not what a multi-objective optimiser normally produces, and a reader is entitled to treat it as a reason for caution. Three factors plausibly contribute: the Intel Lab stream has only eight attributes, so the subset space is small enough that a population-based search can approach exhaustive coverage where sequential filters cannot; the baselines were run at their default settings rather than tuned per stream; and latency and memory were measured on the framework's own operating point rather than swept. The 128-feature comparison of Table 8, where EMLF is the slowest of the eight methods, is the more representative result and is the one on which the contribution should be judged.

The comparison against the streaming feature-selection literature also places the contribution correctly. OSFS and Fast-OSFS[28], SAOLA and group-SAOLA[29], OGFS[30] and α-investing[27] all process features sequentially under a fixed criterion. Their advantage is speed and their limitation is that the criterion cannot itself adapt. EMLF pays roughly a two-fold runtime penalty to search the subset space jointly and to retune its own operators; whether that trade is worthwhile depends on how fast relevance drifts in the target application, which is an empirical question this study does not settle across domains.

Threats to Validity

Construct Validity

- The derived target is not yet specified. The Intel Lab corpus carries no annotations, so the binary label used here is derived from the stream. Until the derivation rule is stated (Section 3.3), the meaning of every accuracy figure on this stream is underdetermined. This is the most consequential open item in the paper.
- Feature reduction is defined by a threshold. The 38% figure is the proportion of attributes below a 50% selection frequency. A different threshold would give a different rate; the sensitivity of the headline figure to that choice has not been tested.
- No feature-selection stability measure. The natural evidence for an adaptation claim is the stability of successive selections, measured by the Kuncheva index[37] or pairwise Jaccard similarity between consecutive subsets. Neither is reported.

Internal Validity

- Dominance on every axis of Table 6. Winning on accuracy, F1, latency, memory and reduction simultaneously against all five baselines is an unusual profile for a multi-objective method. Baselines were run at default settings; per-stream tuning of the comparators would be the appropriate control and was not performed.
- Detection delay not logged. Table 10 reports recovery times, which presuppose detection events, but the ADWIN detection delay is not reported. Algorithm 2 produces it; it must be recorded.
- No hyperparameter search. Population size, generation cap, crossover and mutation probabilities, γ, α and χ were fixed from preliminary trials without recorded validation scores. The value

$\chi = 0.01$ in Eq. 15 places almost all weight on the current mini-batch and almost none on the accumulated history, which is an unusual setting for a forgetting factor and should be verified against the implementation.

- No sensitivity analysis. The fitness weights w_1, w_2 and w_3 were not swept, so the Pareto front of Table 12 collapses to a single point and no accuracy–cost trade-off curve is available.

External Validity

- Two datasets, one hardware platform. Evaluation covers the Intel Lab stream and one 128-feature benchmark, on a single machine. Five or more heterogeneous streams would be required before a general claim about streaming feature selection could be made, and a Friedman–Nemenyi analysis across datasets cannot be run on two.
- The high-dimensional benchmark is not yet identified. Section 4.3 carries a field for its name, source and licence. Until that is supplied, Tables 7 and 8 are not reproducible.
- Filter comparison at one dimensionality. The wrapper is compared against a filter baseline on a single 128-feature stream. Whether the wrapper's advantage persists as dimensionality grows further is untested.

Statistical Conclusion Validity

- Seed variation is not dataset variation. The 30 runs differ only in random seed on a fixed stream, so the standard deviations in Table 6 characterise algorithmic stability, not generalisation. The very large effect sizes in Table 9 follow directly from these small SDs and should be read in that light.
- No cross-dataset omnibus test. Holm correction is applied across the five within-dataset comparisons, but the Friedman test with Nemenyi post-hoc that Demšar[38] recommends for comparing classifiers requires multiple datasets and is not available here.

Limitations and Practical Implications

The evolutionary wrapper carries a real computational cost: 90 of the 135 ms per mini-batch, or 66.7% of the budget (Table 3). At 720 MB memory and 135 ms latency the framework is not deployable on ultra-low-resource IoT endpoints, and is better positioned at the gateway or edge-server tier. The framework also depends on parameter settings for the evolutionary operators and the meta-controller, which may require domain expertise to transfer. Performance under noise characteristics or missing-data patterns different from those of the evaluated streams is untested, and the incremental learner's assumption that feature distributions are locally stable over short intervals may not hold in highly volatile environments. Practically, dynamic relevance selection can raise model accuracy while lowering computational cost in resource-constrained IoT, industrial monitoring and financial prediction settings. Because adaptation is automatic, the framework absorbs drift without frequent manual retuning, which improves operational reliability. Feature reduction additionally improves interpretability and reduces the latency of downstream decisions, which matters for anomaly detection, predictive maintenance and adaptive control.

Conclusion

This paper presented an evolutionary meta-learning framework for dynamic feature selection in streaming prediction. The framework encodes candidate subsets as binary chromosomes, scores them under a normalised multi-objective fitness function over accuracy, latency and memory, adapts its own operator probabilities through an exponentiated-gradient meta-update, and re-seeds its population on an explicit ADWIN drift alarm. On the Intel Lab Sensor stream under a prequential protocol over 30 runs, EMLF attained 92.60 $\pm$ 0.32% accuracy, exceeding the strongest baseline by 2.60 points (95% CI [2.37, 2.83], Holm-adjusted $p < .001$, $d = 5.78$), at 135 ms latency and a 37.5% feature reduction rate. On a 128-feature stream it attained 96.27% accuracy against 93.42% for the strongest streaming feature-selection comparator, while retaining the smallest subset and incurring the highest runtime of the eight methods compared. The ablation attributes 7.8 accuracy points to evolutionary search and 6.2 to the meta-controller. The contribution is bounded by the evidence. Two streams and one hardware configuration do not establish generality; the derived prediction target on the Intel Lab stream requires specification; and the dominance profile of Table 6 is presented with the caution it warrants rather than as a headline. Future work should extend the evaluation to five or more heterogeneous streams with a Friedman–Nemenyi analysis, sweep the fitness weights to produce a genuine Pareto front, report selection stability over the stream, and log ADWIN detection delay alongside recovery time. Deployment in a live industrial or edge environment remains the decisive test of scalability and practicality.

Declarations

Ethical approval: This article does not contain any studies with human or animal subjects performed by any of the authors. The clinical patient-consent statement present in earlier versions has been removed; it does not apply to a sensor-network study.

Funding: Not Applicable

Conflict of interest: The authors declare that they have no conflict of interest. (This is the single declaration; the duplicate and inconsistent statement in earlier versions has been removed.)

Data availability: This study uses the publicly available Intel Berkeley Research Lab Sensor dataset. No new data were collected. The second, high-dimensional benchmark is identified from https://www.kaggle.com/datasets/divyansh22/intel-berkeley-research-lab-sensor-data.

Code availability: Source code, random seeds, preprocessing scripts, drift-injection scripts and the exact stream splits are archived at Repository. DOI: 10.24432/C5MK6M. The pseudocode of Algorithms 1 and 2 specifies the procedure independently of the implementation.

Acknowledgement

The authors are thankful to the Deanship of Graduate Studies and Scientific Research at University of Bisha for supporting this work through the Fast-Track Research Support Program.

Author contributions (CRediT)

M. S. Bennet Praba: Conceptualisation, methodology, supervision, writing, review and editing.
L. N. C. Prakash K: Methodology, investigation, formal analysis, writing, original draft.
Manisha Gupta: Investigation, validation, formal analysis, writing, original draft.
Nikhat Parveen: Investigation, data curation, validation, writing, review and editing.
N. V. S. Pavan Kumar: Software, formal analysis, validation, writing, review and editing.
Vishnu Gandhi V: Conceptualisation, methodology, supervision, validation, writing, review and editing.
All authors reviewed and approved the final manuscript.

ORCID iD

M. S. Bennet Praba https://orcid.org/0000-0002-0452-5360
L. N. C. Prakash K https://orcid.org/0000-0002-0084-1331
Manisha Gupta https://orcid.org/0000-0003-0065-703X
Nikhat Parveen https://orcid.org/0000-0003-2939-0025
N V S Pavan Kumar https://orcid.org/0000-0003-1655-8212
Vishnu Gandhi V https://orcid.org/0009-0000-5654-6641

References

[1] Villegas-Ch, W., García-Ortiz, J., & Sánchez-Viteri, S. (2024). Toward intelligent monitoring in IoT: AI applications for real-time analysis and prediction. *IEEE Access*, 12, 40368–40386. https://doi.org/10.1109/ACCESS.2024.3376707

[2] Raju, K. N., Sudha, V., Kshirsagar, P. R., Tirth, V., & Rajaram, A. (2025). Intelligent traffic prediction system using hybrid convolutional neural networks for smart cities. *Multimedia Tools and Applications*, 84(27), 31919–31937.

[3] Chen, W., Hussain, W., Cauteruccio, F., & Zhang, X. (2024). Deep learning for financial time series prediction: A state-of-the-art review of standalone and hybrid models. *Computer Modeling in Engineering & Sciences*, 139(1), 187–224.

[4] Sharma, R. (2024). Enhancing industrial automation and safety through real-time monitoring and control systems. *International Journal on Smart & Sustainable Intelligent Computing*, 1(2), 1–20.

[5] Kajornkasirat, S., Sawangwong, C., Puangsuwan, K., Chanapai, N., Phutthamongkhon, W., & Puttinaovarat, S. (2025). Integrating AI-driven predictive analytics in wearable IoT for real-time

health monitoring in smart healthcare systems. *Applied Sciences*, 15(8), 4400. https://doi.org/10.3390/app15084400

[6] Zhao, X., Sun, Y., Li, Y., Jia, N., & Xu, J. (2025). Applications of machine learning in real-time control systems: A review. *Measurement Science and Technology*, 36(1), 012003.

[7] Bhatia, M. P. S., & Sangwan, S. R. (2024). Soft computing for anomaly detection and prediction to mitigate IoT-based real-time abuse. *Personal and Ubiquitous Computing*, 28(1), 123–133.

[8] Rustam, F., Raza, A., Qasim, M., Posa, S. K., & Jurcut, A. D. (2024). A novel approach for real-time server-based attack detection using meta-learning. *IEEE Access*, 12, 39614–39627.

[9] Sharma, P., Khan, J. S. A., Radhika, K., Thatipudi, J. G., Sridevi, K., & Upadhyay, S. (2025). Evolutionary intelligence: A hybrid neural framework for dynamic system innovation. In *2025 International Conference on Electronics and Renewable Systems (ICEARS)* (pp. 1743–1749). IEEE.

[10] Swaminathan, B., Palani, S., & Vairavasundaram, S. (2022). Meta learning-based dynamic ensemble model for crop selection. *Applied Artificial Intelligence*, 36(1), 2145646.

[11] Wu, G., Cong, L., Huang, C., Ju, Y., Jiang, J., & Chen, C. (2025). Meta-learning framework for effective few-shot time series prediction. In *2025 IEEE 5th International Conference on Power, Electronics and Computer Applications (ICPECA)* (pp. 18–22). IEEE.

[12] Noor, K., & Fatima, U. (2024). Meta learning for rapid adaptation in financial time series forecasting. *IEEE Access*, 12, 109327–109340.

[13] Zhang, Y., Li, Y., Zhou, X., Luo, J., & Zhang, Z. L. (2022). Urban traffic dynamics prediction — a continuous spatial-temporal meta-learning approach. *ACM Transactions on Intelligent Systems and Technology*, 13(2), 1–19.

[14] Abdallah, M., Rossi, R. A., Mahadik, K., Kim, S., Zhao, H., & Bagchi, S. (2025). Evaluation-free time-series forecasting model selection via meta-learning. *ACM Transactions on Knowledge Discovery from Data*, 19(3), 1–41.

[15] Monteiro, J. P., Ramos, D., Carneiro, D., Duarte, F., Fernandes, J. M., & Novais, P. (2021). Meta-learning and the new challenges of machine learning. *International Journal of Intelligent Systems*, 36(11), 6240–6272. https://doi.org/10.1002/int.22549

[16] Wang, H., & You, D. (2020). Online streaming feature selection via multi-conditional independence and mutual information entropy. *International Journal of Computational Intelligence Systems*, 13(1), 479–487.

[17] Al Nuaimi, N., & Masud, M. M. (2020). Online streaming feature selection with incremental feature grouping. *WIREs Data Mining and Knowledge Discovery*, 10(4), e1364.

[18] Zhao, J., Li, Q., Hong, Y., & Shen, M. (2024). MetaRockETC: Adaptive encrypted traffic classification in complex network environments via time series analysis and meta-learning. *IEEE Transactions on Network and Service Management*, 21(2), 2460–2476.

[19] Gopakumar, M., & Sankaran, S. (2024). Enhancing security in dynamic IoT networks through incremental meta-learning. In *2024 IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS)* (pp. 1–6). IEEE.

[20] Liao, Z., Liu, Y., & Zhao, J. (2023). Meta-learning-based multi-objective PSO model for dynamic scheduling optimization. *Energy Reports*, 9, 1227–1236.

[21] Zou, H., Luo, Y., Lu, X., Jiang, H., & Xie, L. (2015). A mutual information based online access point selection strategy for WiFi indoor localization. In *2015 IEEE International Conference on Automation Science and Engineering (CASE)* (pp. 180–185). IEEE.

[22] Zhao, H., Yuen, P. C., & Kwok, J. T. (2006). A novel incremental principal component analysis and its application for face recognition. *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, 36(4), 873–886. https://doi.org/10.1109/TSMCB.2006.870645

[23] Wang, C., Ren, M., Yu, X., Lin, Y., & Li, S. (2022). Hierarchical online streaming feature selection based on adaptive ReliefF. In *2022 12th International Conference on Information Technology in Medicine and Education (ITME)* (pp. 613–617). IEEE.

[24] Gomes, H. M., Bifet, A., Read, J., Barddal, J. P., Enembreck, F., Pfharinger, B., Holmes, G., & Abdessalem, T. (2017). Adaptive random forests for evolving data stream classification. *Machine Learning*, 106(9), 1469–1495. https://doi.org/10.1007/s10994-017-5642-8

[25] Ayyıldız, E., & Murat, M. (2024). A lasso regression-based forecasting model for daily gasoline consumption: Türkiye case. *Turkish Journal of Engineering*, 8(1), 162–174.

[26] Beyazit, E., Alagurajah, J., & Wu, X. (2019). Online learning from data streams with varying feature spaces. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33(1), 3232–3239. https://doi.org/10.1609/aaai.v33i01.33013232

[27] Zhou, J., Foster, D. P., Stine, R. A., & Ungar, L. H. (2006). Streamwise feature selection. *Journal of Machine Learning Research*, 7, 1861–1885.

[28] Wu, X., Yu, K., Ding, W., Wang, H., & Zhu, X. (2013). Online feature selection with streaming features. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(5), 1178–1192. https://doi.org/10.1109/TPAMI.2012.197

[29] Yu, K., Wu, X., Ding, W., & Pei, J. (2016). Scalable and accurate online feature selection for big data. *ACM Transactions on Knowledge Discovery from Data*, 11(2), 16. https://doi.org/10.1145/2976744

[30] Wang, J., Zhao, Z.-Q., Hu, X., Cheung, Y.-M., Wang, M., & Wu, X. (2013). Online group feature selection. In *Proceedings of the 23rd International Joint Conference on Artificial Intelligence (IJCAI)* (pp. 1757–1763).

[31] Gama, J., Medas, P., Castillo, G., & Rodrigues, P. (2004). Learning with drift detection. In *Advances in Artificial Intelligence — SBIA 2004* (pp. 286–295). Springer. https://doi.org/10.1007/978-3-540-28645-5_29

[32] Gama, J., Žliobaitė, I., Bifet, A., Pechenizkiy, M., & Bouchachia, A. (2014). A survey on concept drift adaptation. *ACM Computing Surveys*, 46(4), 1–37. https://doi.org/10.1145/2523813

[33] Bifet, A., & Gavaldà, R. (2007). Learning from time-changing data with adaptive windowing. In *Proceedings of the 2007 SIAM International Conference on Data Mining* (pp. 443–448). https://doi.org/10.1137/1.9781611972771.42

[34] Srinivas, M., & Patnaik, L. M. (1994). Adaptive probabilities of crossover and mutation in genetic algorithms. *IEEE Transactions on Systems, Man, and Cybernetics*, 24(4), 656–667. https://doi.org/10.1109/21.286385

[35] Eiben, A. E., Hinterding, R., & Michalewicz, Z. (1999). Parameter control in evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 3(2), 124–141. https://doi.org/10.1109/4235.771166

[36] Fialho, Á., Da Costa, L., Schoenauer, M., & Sebag, M. (2010). Analyzing bandit-based adaptive operator selection mechanisms. *Annals of Mathematics and Artificial Intelligence*, 60(1), 25–64. https://doi.org/10.1007/s10472-010-9213-y

[37] Kuncheva, L. I. (2007). A stability index for feature selection. In *Proceedings of the 25th IASTED International Multi-Conference: Artificial Intelligence and Applications* (pp. 390–395).

[38] Demšar, J. (2006). Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, 7, 1–30.

[39] Intel Berkeley Research Lab Sensor Data. Kaggle.